NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ALGORITHMS FOR COMPUTER AIDED
DESIGN OF DIGITAL FILTERS

by

Thomas H. Rich

December 1988

Thesis Advisor: Robert Strum

Approved for public release; distribution is unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL *(If applicable)* 62	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000	7b ADDRESS *(City, State, and ZIP Code)* Monterey, California 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)*	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO · PROJECT NO · TASK NO · WORK UNIT ACCESSION NO

11 TITLE *(Include Security Classification)* ALGORITHMS FOR COMPUTER AIDED DESIGN OF DIGITAL FILTERS

12. PERSONAL AUTHOR(S)
RICH, Thomas H.

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15 PAGE COUNT
Master's Thesis	FROM ________ TO ______	1988 December	57

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	polynomial substitution; analog-digital design; digital-digital design

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The algorithms necessary for the computer aided design of digital filters from lowpass prototypes has been developed and compared, then implemented in a computer program to aid in the instruction of digital filter design.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL STRUM, Robert D.	22b TELEPHONE *(Include Area Code)* 408-646-3451 22c OFFICE SYMBOL 62St

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 UNCLASSIFIED

Approved for public release; distribution is unlimited

ALGORITHMS FOR COMPUTER AIDED DESIGN OF DIGITAL FILTERS

by

Thomas H. Rich
Captain, United States Marine Corps
BSME, Rensselaer Polytechnic Institute, 1981

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

December 1988

ABSTRACT

The algorithms necessary for the computer aided design of digital filters from lowpass prototypes have been developed and compared, then implemented in a computer program to aid in the instruction of digital filter design.

TABLE OF CONTENTS

I. DEVELOPMENT OF GENERAL ALGORITHMS 1

A. INTRODUCTION 1

B. GENERAL DISCUSSION 4

II. DEVELOPMENT OF SPECIFIC TRANSFORMATIONS 13

A. INTRODUCTION 13

B. BILINEAL TRANSFORMATION 13

C. LOWPASS TO LOWPASS DIGITAL TRANSFORMATION 16

D. LOWPASS TO HIGHPASS DIGITAL TRANSFORMATION 19

E. LOWPASS TO BANDSTOP DIGITAL TRANSFORMATION 20

F. LOWPASS TO BANDPASS DIGITAL TRANSFORMATION 23

G. LOWPASS TO LOWPASS ANALOG TRANSFORMATION 24

H. LOWPASS TO HIGHPASS ANALOG TRANSFORMATION 25

I. LOWPASS TO BANDPASS ANALOG TRANSFORMATION 26

J. LOWPASS TO BANDSTOP ANALOG TRANSFORMATION 28

K. GENERAL OBSERVATIONS 29

L. IMPLEMENTATION 32

LIST OF REFERENCES 34

APPENDIX: DFCADD FORTRAN PROGRAM 35

INITIAL DISTRIBUTION LIST 52

I. DEVELOPMENT OF GENERAL ALGORITHMS

A. INTRODUCTION

The procedure for the design of lowpass, highpass, bandpass, and bandstop digital filters from lowpass prototypes is well established, and is based on transforming the prototype frequency response to meet desired characteristics. This transformation can be accomplished in either the analog (s) domain or the digital (z) domain (Figure 1).

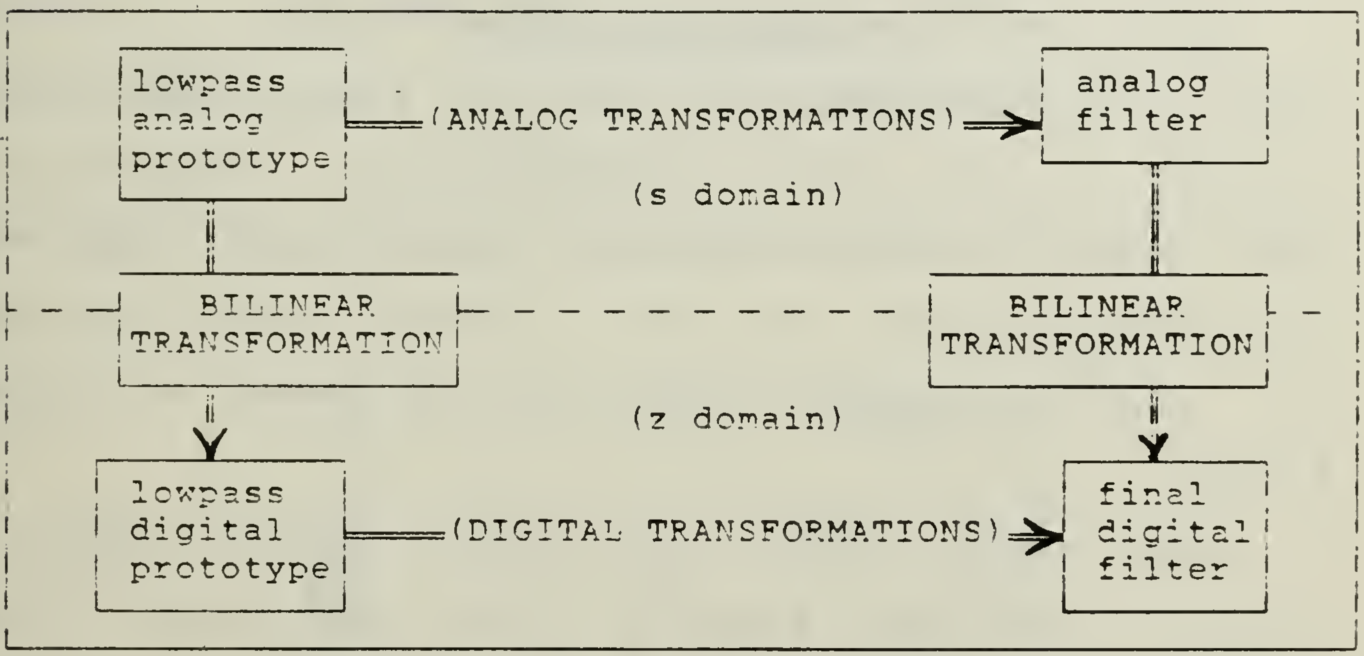

Figure 1. Two methods for digital filter design. Note that each of the horizontal arrows actually represent four different transformations:
1) lowpass prototype to lowpass filter
2) lowpass prototype to highpass filter
3) lowpass prototype to bandpass filter
4) lowpass prototype to bandstop filter

Therefore, the two classical design techniques, analog-digital and digital-digital design, are based on doing the frequency response transformations in their respective analog or digital domains. Since the theory for the standard filter types (Butterworth, Chebyshev, and elliptic) is developed in the analog or 's' domain, the two methods for designing digital filters differ basically in the sequence of the design steps. In the analog-digital method the analog prototype is transformed and then digitized, while in the digital-digital method the analog prototype is digitized and then transformed.

Each of the transformations (arrows in Figure 1) involves substituting a function of 's' or 'z' for the appropriate variable in the filter transfer function. The purpose of this thesis is to develop efficient algorithms to carry out these transformations and then to implement these algorithms in a computer program to aid in the design of digital filters.

Computer-aided design of digital filters offers a significant advantage since it allows the digital filter coefficients to be calculated quickly and accurately. The accuracy of the filter coefficients is an especially significant point since the frequency response of the final filter is very sensitive to small inaccuracies in the calculated coefficients. As an example, compare the frequency response for the filters in Figure 2 and 3. Figure

2 shows the frequency response magnitude for a 1/2-dB ripple Chebyshev bandpass filter, and Figure 3 shows the frequency response when one of the coefficients has been changed by 1/1000th.

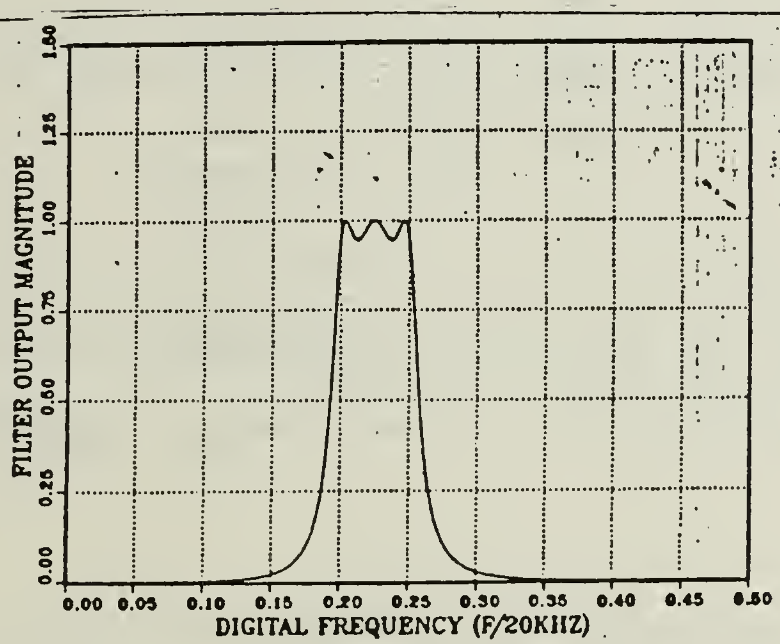

Figure 2. Chebyshev Bandpass Filter

$$H(z) = \frac{.00229(z^2 - 1)^3}{z^6 - .87775z^5 + 2.80076z^4 - 1.5389z^3 + 2.46277z^2 - .67494z + .67529}$$

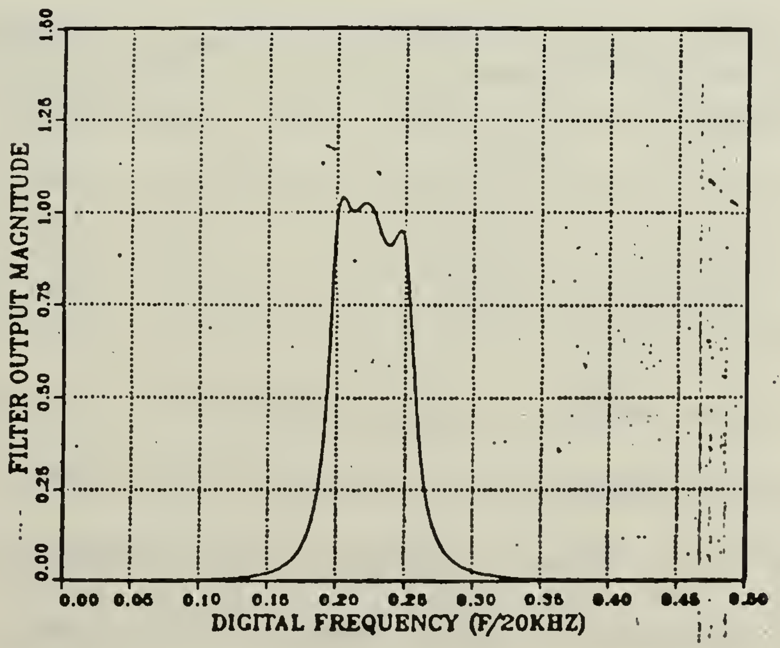

Figure 3. Same Filter With Denominator Coefficient of z changed to -.67594

B. GENERAL DISCUSSION

Each of the transformations involves substituting a ratio of two polynomials of s or z into the filter transfer function which is also a ratio of two polynomials in s or z, namely

$$H(y) = \left. \frac{Y(y)}{U(y)} \right|_{y = \frac{p(x)}{q(x)}} \tag{1.1}$$

The variables 'x' and 'y' can be either 's' or 'z' depending on the transformation to be performed. Therefore, we have either

$$H(y) = \left. \frac{a_0 + a_1 y + a_2 y^2 + \ldots + a_m y^m}{b_0 + b_1 y + b_2 y^2 + \ldots + b_n y^n} \right|_{y = \frac{p(x)}{q(x)}} \tag{1.2}$$

or

$$H(x) = \frac{a_0 + a_1 \dfrac{p(x)}{q(x)} + a_2 \dfrac{(p(x))^2}{(q(x))^2} + \ldots + a_m \dfrac{(p(x))^m}{(q(x))^m}}{b_0 + b_1 \dfrac{p(x)}{q(x)} + b_1 \dfrac{(p(x))^2}{(q(x))^2} + \ldots + b_n \dfrac{(p(x))^n}{(q(x))^n}} \,. \tag{1.3}$$

In general, the order of the numerator 'm', is less than the order of the denominator 'n' (except for the transformations in the digital domain where the result of the bilinear transformation is to make n=m). We can generalize the

results, however, by assuming that the order or the numerator polynomial $Y(x)$ is equal to the order of the denominator polynomial $U(x)$ and making the additional coefficients $a_{m+1}, a_{m+2}, \ldots, a_n$ equal to zero. Then multiplying both the numerator and denominator of the above equation by $(g(x))^n$ gives

$$H(x) = \frac{a_0(g(x))^n + a_1 p(x)(g(x))^{n-1} + \ldots + a_n(p(x))^n}{b_0(g(x))^n + b_1 p(x)(g(x))^{n-1} + \ldots + b_n(p(x))^n} \tag{1.4}$$

or, in a different form,

$$H(x) = \frac{\sum_{i=0}^{i=n} a_i \ p(x)^i g(x)^{n-i}}{\sum_{i=0}^{i=n} b_i \ p(x)^i g(x)^{n-i}} \quad . \tag{1.5}$$

Now with the numerator and denominator of the same form, we can use the same transformation for both the numerator and denominator polynomials of the filter transfer function, namely,

$$a'(x) = a_0' + a_1'x + a_2'x^2 + \ldots + a_n'x^n = \sum_{i=0}^{i=n} a_i \ p(x)^i g(x)^{n-i} \tag{1.6}$$

In Equation 1.6, the a_i are the coefficients of the old numerator or denominator polynomials and a_i' are the new, transformed numerator or denominator polynomial coefficients. Since the new coefficients a_i', are a linear combination of

the old coefficients a_i, and the coefficients of p(x) and q(x), this suggests the use of a matrix transformation;

$$\mathbf{a}' = \mathbf{a}\mathbf{A}(n) \tag{1.7}$$

where

a is the coefficient row vector for the old numerator or denominator polynomial of length=n+1.

$$\mathbf{a} = [a_0 \;\; a_1 \;\; a_2 \;\; . \;\; . \;\; . \;\; a_n]$$

a' is the coefficient row vector for the new numerator or denominator polynomial of length=n'+1.

$$\mathbf{a}' = [a'_0 \;\; a'_1 \;\; a'_2 \;\; . \;\; . \;\; . \;\; a_n']$$

A(n) is the transformation matrix with dimensions (n+1)x(n'+1). The elements of **A**(n) depend upon the coefficients of p(x) and q(x). Note that if the rows and columns of **A**(n) are numbered 0 to n', then

$$a_j' = \sum_{i=0}^{i=n} a_i A(n)_{ij} \tag{1.8}$$

(where $A(n)_{ij}$ denotes the element in row i, column j of matrix **A**(n).)

n is the order of the transfer function before the transformation

n' is the order of the transfer function after the transformation. Note that n' equals n times the highest order of p(x) and q(x).

For example, when n=1 and n'=2, p(x) and q(x) are second order and Equation 1.7 becomes,

$$\begin{bmatrix} a_0 & a_1 \end{bmatrix} \begin{bmatrix} A(2)_{00} & A(2)_{01} & A(2)_{02} \\ A(2)_{10} & A(2)_{11} & A(2)_{12} \end{bmatrix} = \begin{bmatrix} a0' & a1' & a2' \end{bmatrix}. \tag{1.9}$$

In the case of the lowpass digital, highpass digital, and bilinear transformations, the two functions p(x) and g(x) are first order. Professor P.H. Moose of the Naval Postgraduate School of Monterey has developed the matrix **A**(n) for these two cases by expanding Equation 1.6 using the binomial theorem (Reference 1). This method cannot be used for transformations involving higher order polynomials such as the bandpass and bandstop transformations and it is difficult to develop because intricate manipulation of the binomial coefficients are involved. The binomial expansion approach, therefore, was not pursued.

A general method has been found that develops the required transformation matrices in an iterative manner. In other words, the matrix used to transform a numerator or denominator filter polynomial of order n is developed from the matrix for the polynomial of order n-1. This approach is easy to implement in a computer program and can be used for any of the polynomial substitutions needed. The next chapter is concerned with developing the transformation matrices for each specific transformation. However, as an introduction to the method used, consider the generic case where both p(x) and g(x) are second-order polynomials. Since all of the analog and digital transformations involve substitutions with polynomials of second order or less, this second-order generic case will be all that is needed to develop the specific transformations later.

With $p(x)=ax^2 + bx + c$, $g(x) = dx^2 + ex + f$, and substituting into Equation 1.6 for the first order case of n=1 (remembering that with p(x) and g(x) of second order, the resulting polynomial will be second order), we get

$$a'(x) = a_0' + a_1'x + a_2'x^2 = a_0(dx^2 + ex + f) + a_1(ax^2+bx+c) \tag{1.10}$$

Using Equation 1.8 or 1.9 it is easy to pick out the matrix coefficients $A(1)_{ij}$ (i=0,1; j=0,1,2) for the first order generic case. The coefficients are,

$$\mathbf{A}(1) = \begin{bmatrix} f & e & d \\ c & b & a \end{bmatrix} \tag{1.11}$$

Now with n=2 (n'=4) substituting into Equation 1.6 gives,

$$a'(x) = a_0' + a_1'x + a_2'x^2 + a_3'x^3 + a_4'x^4 \tag{1.12}$$

$$= a_0(dx^2+ex+f)^2 + a_1(ax^2+bx+c)(dx^2+ex+f) + a_2(ax^2+bx+c)^2$$

and the matrix **A**(2) is;

$$\mathbf{A}(2) = \begin{bmatrix} f^2 & 2ef & 2df+e^2 & 2de & d^2 \\ fc & ec+bf & cd+fa+eb & db+ea & da \\ c^2 & 2bc & 2ac+b^2 & 2ab & a \end{bmatrix} \tag{1.13}$$

To see how this matrix is derived from the previous one, consider the first row of the matrices in Equations 1.11 and 1.13. For both matrices, this row is multiplied by the a_0 coefficient in the old polynomial. For n=1 we have $a_0 p(x)$

and we pick out the coefficients of the powers of x (x^j) to get the matrix elements $A(n)_{0j}$. For n=2, we have $a_0(p(x))^2$ and we do the same thing only now we have the additional factor of p(x). The matrix elements can be calculated by hand in this fashion but it requires a lot of algebra. A simpler method is to use the matrix elements of the previous matrix and get the appropriate elements by convolving coefficients. For example, consider the sequence of elements in row 0 (the top row) of **A**(1), namely

$$f \quad e \quad d \tag{1.14}$$

To get the elements for row 0 of **A**(2), we need to multiply (convolve coefficients) by another p(x). Therefore, with rows numbered 0 to n, and columns number 0 to n', we have the following.

For row 0 column 0, the appropriate matrix element is

$$A(2)_{00} = \begin{array}{ccccc} d & e & f & & \\ & & f & e & d \end{array} = f^2 \tag{1.15}$$

For row 0 column 1, the appropriate matrix element is

$$A(2)_{01} = \begin{array}{cccc} d & e & f & \\ & f & e & d \end{array} = 2fe \tag{1.16}$$

For row 0 column 2, the appropriate matrix element is

$$A(2)_{02} = \begin{array}{ccc} d & e & f \\ f & e & d \end{array} = 2fd+e^2 \tag{1.17}$$

For row 0 column 3, the appropriate matrix element is

$$A(2)_{03} = \begin{array}{cccccc} & & d & e & f & \\ f & e & d & & & \end{array} = 2de \tag{1.18}$$

For row 0 column 4, the appropriate matrix element is

$$A(2)_{04} = \begin{array}{cccccc} & & & d & e & f \\ f & e & d & & & \end{array} = d^2 \tag{1.19}$$

This convolution of the coefficients of the old matrix with the coefficients of p(x) can be carried out for every row of the old matrix to obtain all the rows of the new matrix except the last row. In the case of the last row, we are multiplying by an additional g(x) rather than an additional p(x) and so must obtain the last row of the new matrix by convolving the coefficients of the last row of the old matrix with the coefficients of g(x).

In summary we have,

<u>(i) for row 0 to row n-1:</u>

-row i of **A**(n) = row i of **A**(n-1) * coefficients of p(x) (1.20)

<u>(ii) for row n:</u>

-row n of **A**(n) = row n-1 of **A**(n-1) * coefficients of g(x) (1.21)

where * indicates linear convolution of the coefficients.

To see how this method of convolving coefficients works, consider multiplying two polynomials:

$$f1(x) = f2(x)f3(x) \tag{1.22}$$

Since 'x' is the independent variable, no matter what letter we use, the polynomials will remain the same. Substituting 'z^{-1}' for 'x' in the above equation gives us,

$$f1(z^{-1}) = f2(z^{-1})f3(z^{-1}) \tag{1.23}$$

and we now have the familiar form where we can either multiply the two functions of 'z' or convolve the sequences found by taking the inverse z-transform. For $f2(z^{-1})$ and $f3(z^{-1})$ we find

$$\begin{aligned} f2(z^{-1}) &= a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n} \\ f3(z^{-1}) &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m} \end{aligned} \tag{1.24}$$

It is now clear that we can find the coefficients of the product f1(x) by convolving the sequence $\{a_n\}$ with the sequence $\{b_m\}$ to produce the sequence $\{c_q\}$ where,

$$\{a_n\} = \{a_0, a_1, a_2, a_3, \ldots, a_n\}$$

$$\{b_m\} = \{b_0, b_1, b_2, b_3, \ldots, b_m\}$$

and with $q = m + n$ (1.25)

$$\{c_q\} = \{c_0, c_1, c_2, c_3, \ldots, c_q\}$$

Then by taking the z-transform of the sequence $\{c_q\}$ and substituting 'x' for 'z^{-1}' we get,

$$f1(x) = c_0 + c_1 x + \ldots + c_q x^q \tag{1.26}$$

We now have a simple straight-forward method of developing the necessary transformation matrices for digital filter design by picking out the first-order transformation matrix using Equations 1.6 and 1.11, and then using the convolution of coefficients method iteratively on the rows of each successive transformation matrix with Equations 1.20 and 1.21 until the appropriate matrix **A**(n) has been generated. The next chapter will develop the transformation matrices for each of the following transformations:

- bilinear transformation
- digital lowpass prototype to digital lowpass filter
- digital lowpass prototype to digital highpass filter
- digital lowpass prototype to digital bandpass filter
- digital lowpass prototype to digital bandstop filter
- analog lowpass prototype to analog lowpass filter
- analog lowpass prototype to analog highpass filter
- analog lowpass prototype to analog bandpass filter
- analog lowpass prototype to analog bandstop filter.

We will see that some of these transformations are similar and consequently, we will not have to develop separate transformation matrices for each case.

II. DEVELOPMENT OF SPECIFIC TRANSFORMATIONS

A. INTRODUCTION

The individual transformations are discussed in the following sections. The specific form of the transformations used in each case are from Reference 2. Since the final result will be to develop a computer program to assist in the calculation of digital filter coefficients, confining the specific transformations to the form in Reference 2, will allow the computer program to be used in conjunction with Reference 2 as an aid in designing digital filters.

B. BILINEAR TRANSFORMATION

The bilinear transformation may be used to transform an analog transfer function, $H(s)$, to a digital transfer function $H(z)$. The transformation from $H(s)$ to $H(z)$ is described by

$$H(z) = H(z)\Big|_{s=\frac{z-1}{z+1}} \tag{2.1}$$

Referring to Equation 1.1, $p(x)=z-1$ and $g(x)=z+1$. Since both of the transformation polynomials $p(x)$ and $g(x)$ are first order, the bilinear transformation will not change the order of the transfer function. Therefore, for an analog transfer function of order 'n', the transformation matrix $\mathbf{A}(n)$ will be

(n+1) by (n+1). From Equation 1.6, the numerator or denominator polynomial of the transfer function H(z) is given by

$$a'(z) = a_0' + a_1'z + \ldots + a_n'z^n = \sum_{i=0}^{i=n} a_i (z-1)^i (z+1)^{n-i} \qquad (2.2)$$

where the unprimed coefficients, a_i, come from the analog transfer function's numerator or denominator polynomial. In matrix form, with n=1 and referring to Equation 1.11, the first order matrix **A**(1) for the bilinear transformation is

$$\mathbf{A}(1) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \qquad (2.3)$$

Using the convolution of coefficients method developed in Chapter 1, the transformation matrix **A**(n+1) can be developed from the matrix **A**(n). The results for the bilinear transformation matrices **A**(1) through **A**(5) are given in Table 1.

TABLE 1

FIRST-ORDER TO FIFTH-ORDER BILINEAR TRANSFORMATION MATRICES

$$\mathbf{A}(1) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$\mathbf{A}(2) = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\mathbf{A}(3) = \begin{bmatrix} 1 & 3 & 3 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$\mathbf{A}(4) = \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ 1 & 0 & -2 & 0 & 1 \\ -1 & 2 & 0 & -2 & 1 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}$$

$$\mathbf{A}(5) = \begin{bmatrix} 1 & 5 & 10 & 10 & 5 & 1 \\ -1 & -3 & -2 & 2 & 3 & 1 \\ 1 & 1 & -2 & -2 & 1 & 1 \\ -1 & 1 & 2 & -2 & -1 & 1 \\ 1 & -3 & 2 & 2 & -3 & 1 \\ -1 & 5 & -10 & 10 & -5 & 1 \end{bmatrix}$$

Referring to Table 1 and keeping in mind the convolution of coefficients method that produced the matrices, the following relations are apparent and will **be useful** in computer implementation.

$$A(n)_{0j} \text{ (the first row)} = \binom{n}{j} \quad \text{for } j=0,1,\ldots,n \tag{2.4}$$

where $\binom{n}{j}$ indicates the binomial coefficient

$$A(n)_{i0} \text{ (the first column)} = (-1)^{i} \quad \text{for } i=0,1,\ldots,n \tag{2.5}$$

$$A(n)_{in} \text{ (the last column)} = 1 \quad \text{for } i=0,1,\ldots,n \tag{2.6}$$

$$A(n)_{(n-i)j} = (-1)^{n-j} A(n)_{ij} \quad \text{for } i=0,1\ldots,\text{integer}(n/2)+1 \tag{2.7}$$

$$A(n)_{ij} = A(n-1)_{ij} + A(n-1)_{i(j-1)} \quad \text{for } \begin{array}{l} i=0,1,\ldots,n-1 \\ j=1,2,\ldots,n-1 \end{array} \tag{2.8}$$

Notice that the values of the matrix elements for the bilinear transformation are whole numbers. This is important from a numerical accuracy standpoint since whole numbers can be expressed precisely in floating point form. This means that the bilinear transformation will cause a loss of accuracy of the original coefficients only because of the n+1 multiplications and additions used to calculate each new digital coefficient.

C. LOWPASS TO LOWPASS DIGITAL TRANSFORMATION

The lowpass to lowpass digital transformation is used to transform a lowpass prototype digital filter **H(z) into** a lowpass digital filter H(z). The polynomial substitution

used is

$$H(z) = H_p(z)\Big|_{z = \frac{z-\alpha}{1-\alpha z}} \qquad (2.9)$$

where the subscript p indicates the prototype digital filter transfer function. The design constant α is determined [References 2 and 3] by

$$\alpha = \frac{\sin(\Theta_c/2 - \Theta_c'/2)}{\sin(\Theta_c/2 - \Theta_c''/2)} \qquad (2.10)$$

with Θ_c = the prototype critical frequency (usually pi/2) and Θ_c'= the desired critical frequency of the lowpass filter. Again, referring to Equation 1.1 with $p(x) = z-\alpha$ and $g(z) = 1-\alpha z$, the lowpass digital transformation matrix will not change the order of the transfer function and the transformation matrix **A**(n) will be (n+1) by (n+1). Using Equations 1.6 and 1.11 as before, the first order transformation matrix **A**(1) for the lowpass digital transformation is:

$$A(1) = \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix} \qquad (2.11)$$

Once again using the convolution of coefficients method, the higher-order transformation matrixes **A**(n) are developed from the lower-order matrix **A**(n-1). The results are shown in Table 2 for the first-order to fifth-order transformation matrices.

TABLE 2
FIRST-ORDER TO FIFTH-ORDER LOWPASS DIGITAL
TRANSFORMATION MATRICES

$$\mathbf{A}(1) = \begin{bmatrix} 1 & -\alpha \\ -\alpha & 1 \end{bmatrix}$$

$$\mathbf{A}(2) = \begin{bmatrix} 1 & -2\alpha & \alpha^2 \\ -\alpha & (1+\alpha^2) & -\alpha \\ \alpha^2 & -2\alpha & 1 \end{bmatrix}$$

$$\mathbf{A}(3) = \begin{bmatrix} 1 & -3\alpha & 3\alpha^2 & -\alpha^3 \\ -\alpha & (1+2\alpha^2) & (-2\alpha-\alpha^3) & \alpha^2 \\ \alpha^2 & (-2\alpha-\alpha^3) & (1+2\alpha^2) & -\alpha \\ -\alpha^3 & 3\alpha^2 & -3\alpha & 1 \end{bmatrix}$$

$$\mathbf{A}(4) = \begin{bmatrix} 1 & -4\alpha & 6\alpha^2 & -4\alpha^3 & \alpha^4 \\ -\alpha & (1+3\alpha^2) & (-3\alpha-3\alpha^3) & (3\alpha^2+\alpha^4) & -\alpha^3 \\ \alpha^2 & (-2\alpha-2\alpha^3) & (1+4\alpha^2+\alpha^4) & (-2\alpha-2\alpha^3) & \alpha^2 \\ -\alpha^3 & (3\alpha^2+\alpha^4) & (-3\alpha-3\alpha^3) & (1+3\alpha^2) & -\alpha \\ \alpha^4 & -4\alpha 3 & 6\alpha^2 & -4\alpha & 1 \end{bmatrix}$$

$$\mathbf{A}(5) = \begin{bmatrix} 1 & -5\alpha & 10\alpha^2 & -10\alpha^3 & 5\alpha^4 & -\alpha^5 \\ -\alpha & (1+3\alpha^2) & (-4\alpha-6\alpha^3) & (6\alpha^2+\alpha^4) & (-4\alpha^3-\alpha^5) & \alpha 4 \\ \alpha^2 & (-2\alpha-3\alpha^3) & (1+6\alpha^2+3\alpha^4) & (-3\alpha-6\alpha^3-\alpha^5) & (3\alpha^2+2\alpha^4) & -\alpha^3 \\ -\alpha^3 & (3\alpha^2+2\alpha^4) & (-3\alpha-6\alpha^2-\alpha^5) & (1+6\alpha^2+3\alpha^4) & (-2\alpha-3\alpha^3) & \alpha 2 \\ \alpha^4 & (-4\alpha^3-\alpha^5) & (6\alpha^2+4\alpha^4) & (-4\alpha-6\alpha^3) & (1+4\alpha^2) & -\alpha \\ -\alpha^5 & 5\alpha^4 & -10\alpha^3 & 10\alpha^2 & -5\alpha & 1 \end{bmatrix}$$

Referring to Table 2, the following relations are observed and will be used in the computer implementation.

$$A(n)_{0j} \quad \text{(the first row)} = \binom{n}{j}(-\alpha)^{j} \quad \text{for } j=0,1, \quad ,n \tag{2.12}$$

$$A(n)_{i0} \quad \text{(the first column)} = (-\alpha)^{i} \quad \text{for } i-0,1,\ldots,n \tag{2.13}$$

$$A(n)_{ij} = A(n)_{(n-i)(n-j)} \quad \text{for } i,j=o,1,\ldots,n \tag{2.14}$$

$$A(n)_{ij} = A(n-1)_{(i-1)(j-1)} + (-\alpha)A(n-1)_{(i-1)j} \tag{2.15}$$
$$\text{for } i=1,2, \quad ,n-1$$
$$j=1,2,\ldots,n$$

Note that the elements of the lowpass digital transformation matrices are not whole numbers as in the bilinear transformation case, but are polynomials of the design constant α. Numerical error is introduced in this case by both the n+1 additions or multiplications, and the error in calculating the matrix elements that multiply the original coefficients.

D. LOWPASS TO HIGHPASS DIGITAL TRANSFORMATION

The lowpass to highpass digital transformation is used to transform a lowpass prototype to a highpass filter. The polynomial substitution for this transformation is:

$$H(z) = H_p(z)\Big|_{z = -\frac{z-\alpha}{1-\alpha z}} \tag{2.16}$$

Since this transformation is the same as that for the lowpass to lowpass digital transformation except for the minus sign,

we can make the substitution of '-z' for 'z' in the original prototype transfer function and then use the same transformation matrices as the lowpass to lowpass case. For the lowpass to highpass transformation, the design constant α is (References 2 and 3)

$$\alpha = \frac{\cos(\Theta_c/2 - \Theta_c'/2)}{\cos(\Theta_c/2 + \Theta_c'/2)} \tag{2.17}$$

with Θc - the prototype critical frequency (usually pi/2) and $\Theta c'$ - the desired critical frequency of the highpass filter.

E. LOWPASS TO BANDSTOP DIGITAL TRANSFORMATION

The polynomial substitution which transforms a digital lowpass prototype to a bandstop digital filter is (References 2 and 3):

$$H(z) = H_p(z) \Bigg|_{z = \frac{z^2 - \frac{2\alpha}{1+k} z + \frac{1-k}{1+k}}{1 - \frac{2\alpha}{1+k} z + \frac{1-k}{1+k} z^2}} \tag{2.18}$$

with

$$\alpha = \frac{\cos(\Theta_u'/2 + \Theta_l'/2)}{\cos(\Theta_u'/2 - \Theta_l'/2)} \tag{2.19}$$

and

$$K = \tan(\Theta_c/2)\tan(\Theta_u'/2 - \Theta_l'/2) \tag{2.20}$$

with once again, Θ_c = the prototype critical frequency (usually pi/2), Θ_u' = the desired upper critical digital

frequency, and Θ_l' = the desired low critical digital frequency.

By letting $b = 2\alpha/(k+1)$ and $c = (1-k)/(1+k)$, Equation 2.18 can be simplified to

$$H(z) = H_p(z)\Big|_{z = \dfrac{z^2 - bz + c}{1 - bz + cz^2}} \qquad (2.21)$$

Again, using Equations 1.6 and 1.11 the first-order transformation matrix is

$$\mathbf{A}(1) = \begin{bmatrix} 1 & -b & c \\ c & -b & 1 \end{bmatrix} \qquad (2.22)$$

Note that since the substitution polynomials are second order, this transformation will double the order of the original prototype filter. Therefore, the transformation matrix $\mathbf{A}(n)$ will be $(n+1)$ by $(2n+1)$. The higher order transformation matricies are again developed recursively and the results for the first-order to third-order transformations are given in Table 3.

TABLE 3
FIRST-ORDER TO THIRD-ORDER BANDSTOP
DIGITAL TRANSFORMATION MATRICES

$$\mathbf{A}(1) = \begin{bmatrix} 1 & -b & c \\ c & -b & 1 \end{bmatrix}$$

$$\mathbf{A}(2) = \begin{bmatrix} 1 & -2b & (b^2+2c) & -2bc & c^2 \\ c & (-b-bc) & (1+b^2+c^2) & (-b-bc) & c \\ c^2 & -2bc & (b^2+2c) & -2c & 1 \end{bmatrix}$$

$$\mathbf{A}(3) = \begin{bmatrix} 1 & -3c & (3b^2+3c) & (-b^3-6bc) & \ldots \\ c & (-2bc-b) & (1+2b^2+b_c+2c) & (-b^3-2b-2bc-2bc^2) & \ldots \\ c^2 & (-2bc-bc^2) & (c^3+2b^2c+2c+b^2) & \ldots & \\ c^3 & -3c^2b & (3b^2c+3c^2) & \ldots & \end{bmatrix}$$

(Redundant entries not shown for **A**(3) due to space limitations. Entries not shown can be found using Equation 2.25)

Notice the increased amount of calculation needed to determine the matrix elements for the bandstop case. Each matrix element is a function of 'b' and 'c' which are themselves functions of the design constants α and k. Table 3 only gives the transformation matrices for first- to third-order. However, to illustrate the amount of calculation required, the matrix element $A(5)_{25}$ is

$$A(5)_{25} = (-12bc^2-6bc^3-8b^3c-6bc-6b^3-3b-3bc^4-6b^3c^2-b^5) \qquad (2.23)$$

Since each coefficient in the final filter **is found** by multiplying n+1 matrix elements by the n+1 original

coefficients and then adding the n+1 products, the accuracy of the design constants and the original coefficients can be critical in the calculation of the final filter **coefficients**.

The symmetry relations for the bandstop **transformation** are:

$$A(n)_{i0} \quad \text{(the first column)} = c^i \quad \text{for } i=0,1,\ldots,n \tag{2.24}$$

$$A(n)_{ij} = A(n)_{(n-i)(2n-j)} \quad \text{for } i=0,1,\ldots,n;\ j=0,1,\ldots,2n \tag{2.25}$$

$$A(n)_{ij} = A(n-1)_{ij} - bA(n-1)_{i(j-1)} + cA(n-1)_{i(j-2)} \quad \text{for } i=0,1,\ldots,n-1;\ j=0,1,\ldots,2n \tag{2.26}$$

F. LOWPASS TO BANDPASS DIGITAL TRANSFORMATION

Again, the polynomial substitution which transforms the lowpass prototype to a bandpass filter is

$$H(z) = H_p(z)\Bigg|_{z = -\dfrac{z^2 - \dfrac{2\alpha k}{k+1} z + \dfrac{k-1}{k+1}}{1 - \dfrac{2\alpha k}{k+1} z + \dfrac{k-1}{k+1} z^2}} \tag{2.27}$$

where in this case $k = \tan(\Theta/2)\cot(\Theta_u'/2 - \Theta_l'/2)$ and α is found from the same expression as the bandstop case. With the simplification of $b=2\alpha k/(k+1)$ and $c=(k-1)/(k+1)$, Equation 2.27 becomes

$$H(z) = H_p(z)\Big|_{z = -\frac{z^2 - bz + c}{1 - bz + cz^2}} \qquad (2.28)$$

Since this is the same form as the bandstop transformation, we can use the matrices previously developed if '-z' is first substituted for 'z' as was done in the lowpass and highpass situation.

G. LOWPASS TO LOWPASS ANALOG TRANSFORMATION

The lowpass to lowpass analog transformation is used to transform an analog lowpass prototype to a lowpass analog filter. The polynomial substitution is a simple one, namely

$$H(s) = H_p(s)\Big|_{s = s/w_c} \qquad (2.29)$$

where w_c is the prewarped critical frequency ($w_c = \tan(\Theta_c/2)$ and Θ_c is the desired digital critical frequency ($\Theta_c = 2(pi)f_c/f_s$). This transformation matrix is found to be

$$A(n) = \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 \\ 0 & 1/w_c & 0 & . & . & . & 0 \\ 0 & 0 & 1/w_c^2 & . & . & . & 0 \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ 0 & 0 & 0 & . & . & . & 1/w_c^n \end{bmatrix}$$

or $A(n) = \mathrm{diag}(1, 1/w_c, 1/w_c^2, \ . \ . \ . \ , 1/w_c^n)$ throughout. (2.30)

Note that in this case, each new filter coefficient depends only on multiplying the old coefficient of s^i by $(w_c)^{-i}$. Therefore, any loss of accuracy due to the transformation is due to this one multiplication and the error introduced in calculation of the matrix element $(w_c)^{-i}$.

H. LOWPASS TO HIGHPASS ANALOG TRANSFORMATION

The polynomial substitution for the lowpass to highpass transformation is

$$H(s) = H_p(s)\Big|_{s = w_c/s} \tag{2.31}$$

Again using Equations 1.6 and 1.11, the first-order transformation matrix is

$$A(1) = \begin{bmatrix} 0 & 1 \\ w_c & 0 \end{bmatrix} \tag{2.32}$$

The method of convolution of coefficients can be used to get the higher-order transformation matrices. However, notice that Equation 2.32 is the same matrix as the lowpass to lowpass transformation with the reciprocal taken of each non zero element, and then each row reversed. This can be shown to be the case in general, so that the lowpass to highpass transformation matrix is given by

$$
A(n) = \begin{bmatrix} 0 & 0 & . & . & . & 0 & 1 \\ 0 & 0 & . & . & . & w_c & 0 \\ . & & & & & & \\ . & & & & & & \\ . & & & & & & \\ 0 & w_c^{\,n-1} & . & . & . & 0 & 0 \\ w_c^{\,n} & 0 & . & . & . & 0 & 0 \end{bmatrix} \qquad (2.33)
$$

Again, each new filter coefficient depends only on one multiplication although the calculation of the matrix element itself will require calculations involving raising the design constant, w_c, to the $(n-1)^{th}$ power where i is the power of s in the final filter transfer function.

I. LOWPASS TO BANDPASS ANALOG TRANSFORMATION

The polynomial substitution for transforming a lowpass analog prototype to a bandpass analog filter is

$$
H(s) = H_p(s)\Big|_{s = \frac{s^2 + wo^2}{Bs}} \qquad (2.34)
$$

where wo is defined as the geometric mean of the passband; $w_o = (w_u w_l)^{1/2}$, and B is the width of the passband, $B = w_u - w_l$. (w_u and w_l are the upper and lower critical frequencies respectively). Again using Equations 1.6 and 1.11 and picking out the appropriate matrix elements, the first-order transformation matrix is given by

$$\mathbf{A}(1) = \begin{bmatrix} 0 & B & 0 \\ w_o^{\,2} & 0 & 1 \end{bmatrix} \tag{2.35}$$

The convolution of coefficients technique is **again** used to generate the higher order transformation matrices with the results for the first-order to third-order matrices listed in Table 4. As with the bandpass digital transformation, the analog bandpass transformation will double the order of the original transfer function since the substitution polynomial is of second order.

TABLE 4
FIRST-ORDER TO THIRD-ORDER BANDPASS
ANALOG TRANSFORMATION MATRICES

$$\mathbf{A}(1) = \begin{bmatrix} 0 & B & 0 \\ w_o^{\,2} & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}(2) = \begin{bmatrix} 0 & 0 & B^2 & 0 & 0 \\ 0 & Bw_o^{\,2} & 0 & B & 0 \\ w_o^{\,4} & 0 & 2w_o^{\,2} & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}(3) = \begin{bmatrix} 0 & 0 & 0 & B^3 & 0 & 0 & 0 \\ 0 & 0 & B^2 w_o^{\,2} & 0 & B^2 & 0 & 0 \\ 0 & Bw_o^{\,4} & 0 & 2Bw_o^{\,2} & 0 & B & 0 \\ w_o^{\,6} & 0 & 3w_o^{\,4} & 0 & 3w_o^{\,2} & 0 & 1 \end{bmatrix}$$

Note that each matrix is slightly over half **empty** and that the nonzero elements are simple functions of the design constants w_o and B.

J. LOWPASS TO BANDSTOP ANALOG TRANSFORMATION

For the analog bandstop transformation, the polynomial substitution is the reciprocal of the bandpass case:

$$H(s) = H_p(s) \Big|_{s = \frac{Bs}{s^2 + w_o^2}} \tag{2.36}$$

Using the same method as before, the first-order transformation matrix is found to be:

$$\mathbf{A}(1) = \begin{bmatrix} w_o^2 & 0 & 1 \\ 0 & B & 0 \end{bmatrix} \tag{2.37}$$

Note that this is the same matrix as the bandpass case but with the rows in reverse order. Using the convolution of coefficients technique the higher-order matrices are developed and the results for the first-order to third-order transformation matrices are shown in Table 5. Notice that the reversal of the order of the rows compared with the bandpass case is true in general. Therefore, once again the matrices are over half empty and nonzero elements are simple functions of B and w_o.

TABLE 5
FIRST-ORDER TO THIRD-ORDER BANDSTOP
ANALOG TRANSFORMATION MATRICES

$$\mathbf{A}(1) = \begin{bmatrix} w_o^2 & 0 & 1 \\ 0 & B & 0 \end{bmatrix}$$

$$\mathbf{A}(2) = \begin{bmatrix} w_o^4 & 0 & 2w_o^2 & 0 & 1 \\ 0 & Bw_o^4 & 0 & B & 0 \\ 0 & 0 & B^2 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}(3) = \begin{bmatrix} w_o^6 & 0 & 2w_o^4 & 0 & 3w_o^{2\;2} & 0 & 1 \\ 0 & Bw_o^4 & 0 & 2Bw_o^2 & 0 & B & 0 \\ 0 & 0 & B^2w_o^2 & 0 & B^2 & 0 & 0 \\ 0 & 0 & 0 & B^3 & 0 & 0 & 0 \end{bmatrix}$$

K. GENERAL OBSERVATIONS

Now that the various transformation matrices for digital filter design have been developed, it is possible to compare the two methods of digital filter design mentioned earlier and depicted in Figure 1. For this comparison, consider an analog lowpass prototype filter transfer function with numerator of order m, and denominator of order n (m less than or equal to n).

First consider the digital to digital direct design method. Transforming the analog prototype to a digital prototype using the bilinear transformation will require m+1

multiplications and additions per digital prototype coefficient for the numerator polynomial, and n+1 multiplications and additions per digital prototype coefficient for the denominator polynomial. Then, each of the digital to digital transformations will require 2(n+1) multiplications and additions since a result of the bilinear transformation is to make n = m. Therefore, going from an analog prototype to a final digital filter will require (3n+m+4) multiplications and additions. If lowpass digital prototypes are developed prior to the design process, this can be reduced to 2(n+1) since then only the digital to digital transformations will be used in designing the final filter.

Now consider the analog design method for the same case of an m^{th}-order numerator and n^{th}-order denominator lowpass prototype transfer function. For the lowpass and highpass case, each new coefficient will require only 1 multiplication and addition. For the bandpass and bandstop transformations a simple expression for the number of calculations per coefficient is difficult to find, however referring to Tables 4 and 5, the number of multiplications and additions per coefficient is always less than half the order of the polynomial to be transformed, on the average. This gives approximately (n+m)/2 for the bandpass to bandstop analog transformations. Finally the analog filter must be converted to a digital filter using the bilinear transformation which

will require (n+m+2) calculations per coefficient for the highpass and lowpass case and 2(n+m+1) for the bandpass and bandstop case. The total number of calculations in using the analog design method then is (n+m+4) for the high/low pass and (2.5(n+m)+2) for the bandpass/stop. These results are summarized in Table 6.

TABLE 6
NUMBER OF MULTIPLICATIONS AND ADDITIONS REQUIRED PER COEFFICIENT TO FIND THE DIGITAL FILTER COEFFICIENTS FROM A LOWPASS ANALOG PROTOTYPE (WITH AN M^{TH} ORDER NUMERATOR AND N^{TH} ORDER DENOMINATOR)

	Analog Design	
	high/low pass	bandpass/stop
analog transformations	2	.5(n+m)
bilinear transformation	n+m+2	2(n+m)+2
Total	n+m+4	2.5(n+m)+2

	Digital Design	
	high/low pass	bandpass/stop
bilinear transformation	n+m+2	n+m+2m
digital transformations	2(n+1)	2(n+1)
Total	3(n+1)+m+1	3(n+1)+m+1

Referring to Table 6, the advantage of one method over the other depends upon what type of filter is being designed and the order of the prototype transfer functions numerator and denominator. Consider designing a fifth-order low/highpass Chebyshev or Butterworth filter. In this case, the order of the numerator polynomial is zero and the analog method will have a slight advantage of 9 multiplications and additions versus 12 for the digital design with precalculated prototypes (and 19 for digital design without precalculated prototypes). However, if a bandpass/stop elliptic filter was being designed with n=5, then m=4 and the direct digital design method (with precalculated prototypes) has an advantage of 12 multiplications and additions compared to approximately 24 for the analog method.

A final consideration in favor of analog design is the complexity of the elements of the transformation matrices. Since the elements of the digital transformation matrices require much more calculation than the elements of the analog transformation matrices or the bilinear transformation (compare Tables 1-5), the problem of roundoff error is much more serious in the case of the digital to digital design method.

L. IMPLEMENTATION

The final step is to implement the algorithms developed previously in a computer program to assist in the teaching of digital filter design. Since the program is intended to be

used for instructional purposes, the numerical considerations discussed earlier are not considered as important as conceptual simplicity. The direct design method was chosen for implementation as the design constants can be found easily without reference to "prewarping." The completed program, called 'DFCADD' (Digital Filter Computer Aided Direct Design), is contained in Appendix A with its own documentation. The program follows the procedures set forth in Chapter 10 of Reference 2. Prototype coefficients are stored within the program for first-order to fifth-order Butterworth and Chebyshev (1/2,1,&2dB ripples) filters. While second-order to sixth-order elliptic lowpass prototype coefficients are calculated within the program for any combination of 1/2,1,2, and 3 dB passband ripple and -20,-30,-40,-50,-60, and -70 dB stopband attenuation, with the help of a program developed by Professor D.E. Kirk of the Naval Postgraduate School and References 4 and 5. As a final option, the user may enter his/her own prototype coefficients for up to a tenth-order analog prototype. After calculating the digital filter coefficients an option exists for the program to create a data file, titled "filter", with the digital filter magnitude and phase as a function of the digital frequency from zero to pi. This data file can then be used with any plotting routine to generate a plot of the frequency response of the filter.

LIST OF REFERENCES

1. Moose, P.H. of the United States Naval Postgraduate School, Monterey, California, Letter to Strum, R.D. and Kirk, D.E., Subject: Algorithm for Digital-Digital Transformations, 5 June 1987.

2. Strum, R.D. and Kirk, D.E., First Principles of Discrete Systems and Digital Signal Processing, pp. 612-701, Addison-Wesley, 1988.

3. Constantinides, A.G., "Spectral Transformations for Digital Filters, "Proceedings of IEEE, p. 117,(August): 1585-1590.

4. Ludeman, L.C., Fundamentals of Digital Signal Processing, pp. 149-160, Harper and Row, 1986.

5. Antoniou, A., Digital Filters: Analysis and Design, pp. 114-125, McGraw-Hill, 1979.

APPENDIX
DFCADD FORTRAN PROGRAM

FILE: DFCADD FORTRAN A

```
C      ***********************************************************
C      * THIS PROGRAM ASSISTS IN THE CALCULATION OF DIGITAL FILTER    *
C      * COEFFICIENTS ACCORDING TO THE PROCEDURE SET FORTH IN CHAPTER *
C      * 10 (SUMMARIZED ON PG 695) OF - FIRST PRINCIPLES OF DISCRETE  *
C      * SYSTEMS AND DIGITAL SIGNAL PROCESSING -  BY R.D. STRUM  AND  *
C      * D.E. KIRK (REFERENCE(2) )                                   *
C      ***********************************************************
       INTEGER ORDER,TYPE,PASS,ORDER2,END,AFORDF
       REAL KPR,LAM,OMEGA(20),V(20),A0(20),B0(20),B1(20)
       COMPLEX Z,NEWZ,ZN,ZD,HOFZN,HOFZD,HOFZ
       DOUBLE PRECISION TEMP,SUMR
       DOUBLE PRECISION R,S0,A01,A02,B01,B02,B11,B12,H0
       DOUBLE PRECISION A03,A04,B03,B04,B13,B14
       DOUBLE PRECISION SFREQ,AFC,DFC,ALPHA,PI,PID4,DFCD2
       DOUBLE PRECISION ALC,AUC,DLC,DUC,RK,B,C,DLCD2,DUCD2
       DOUBLE PRECISION APROT,BILINR,DPROT,LPHPTR,BPBSTR,DFLTR
C      ***********************************************************
C      FORMAT OF ARRAYS:
C         -APROT CONTAINS THE ANALOG PROTOTYPE
C              APROT(NORD,PWR)
C         -BILINR IS THE ANALOG-DIGITAL BILINEAR TRANSFORMATION MATRIX
C              BILINR(ORDER,ROW,COL)
C         -DPROT IS THE DIGITAL PROTOTYPE FROM APROT AND BILINR
C              DPROT(NORD,PWR)
C         -LPHPTR IS THE LP&HP DIGITAL-DIGITAL TRANFORMATION MATRIX
C              LPHPTR(ORDER,ROW,COL)
C         -DFLTR IS THE FINAL DIGITAL FILTER
C              DFLTR(NORD,PWR)
C         -BPBSTR IS THE BP&BS DIGITAL-DIGITAL TRANSFORMATION MATRIX
C              BPBSTR(ORDER,ROW,COL)
C      ***********************************************************
       DIMENSION APROT(2,0:10)
       DIMENSION BILINR(10,0:10,0:10)
       DIMENSION DPROT(2,0:10)
       DIMENSION LPHPTR(10,0:10,0:10)
       DIMENSION BPBSTR(10,0:10,0:20)
       DIMENSION DFLTR(2,0:20)
C      ***********************************************************
       OPEN(UNIT=8,FILE='FILTER DATA',STATUS='UNKNOWN')
C      ***********************************************************
       PI=3.1415926535898D+00
       SPI=3.14159
       PID4=PI/4.0D+00
C      ***********************************************************
C      CLEAR APROT PRIOR TO LOADING
       DO 1 K=1,2
        DO 2 L=0,10
         APROT(K,L)=0.0D+00
2       CONTINUE
1      CONTINUE
C      ***********************************************************
C      ***********************************************************
C      BEGIN INTERACTION
       CALL EXCMS ('CLRSCRN')
       WRITE(6,*) 'THIS IS A PROGRAM TO CALCULATE DIGITAL FILTERS USING'
       WRITE(6,*) ' THE DIRECT DESIGN METHOD.'
       WRITE(6,*)
       WRITE(6,*) '**************************************************'
       WRITE(6,*) '*WARNING-THIS PROGRAM IS NOT USER FRIENDLY!!!  *'
       WRITE(6,*) '* YOU MUST ENTER VALUES OF THE APPROPRIATE TYPE*'
       WRITE(6,*) '* (REAL OR INTEGER), AND PROPER RANGE          *'
       WRITE(6,*) '**************************************************'
       WRITE(6,*)
       WRITE(6,*) '  ** ENTER 1 TO CONTINUE ** '
       READ(5,*)  LOOK
9      CALL EXCMS ('CLRSCRN')
10     WRITE(6,*) 'WHAT TYPE OF FILTER DO YOU WANT?'
       WRITE(6,*) ' (1-4 ALLOW ONLY UP TO 5TH ORDER PROTOTYPES)'
       WRITE(6,*) '      1-BUTTERWORTH '
       WRITE(6,*) '      2-CHEBYSHEV WITH 1/2 DB RIPPLE '
       WRITE(6,*) '      3-CHEBYSHEV WITH 1 DB RIPPLE '
       WRITE(6,*) '      4-CHEBYSHEV WITH 2 DB RIPPLE'
```

```
      WRITE(6,*) '      5-ELLIPTIC (ALLOWS UP TO A 6TH ORDER PROTOTYPE)'
      WRITE(6,*) '        -PASSBAND RIPPLE 0.5,1.0,2.0,OR 3.0 DB'
      WRITE(6,*) '        -STOPBAND ATTENTUATION -20,-30,-40,-50,-60, '
      WRITE(6,*) '             OR -70 DB   '
      WRITE(6,*) '      6-OTHER - ALLOWS YOU TO USE UP TO A 10TH ORDER'
      WRITE(6,*) '         PROTOTYPE, HOWEVER YOU MUST ENTER THE ANALOG'
      WRITE(6,*) '         PROTOTYPE COEFFICIENTS '
      WRITE(6,*)
      WRITE(6,*) '      ** INTEGER 1-6 **'
      READ(5,*) TYPE
C     ****************************************************************
14    CALL EXCMS ('CLRSCRN')
15    WRITE(6,*)
      WRITE(6,*) 'WHAT IS THE ORDER OF THE PROTOTYPE YOU WISH TO USE?'
      WRITE(6,*)
      WRITE(6,*) ' - MUST BE AN INTEGER IN THE RANGE 1-5  UNLESS YOU'
      WRITE(6,*) '     CHOSE TO ENTER YOUR OWN PROTOTYPE (THEN 1-10)'
      WRITE(6,*) '     OR YOU CHOSE AN ELLIPTIC FILTER (THEN 1-6)'
      WRITE(6,*) ' - OR ENTER "0" IF YOU WANT HELP SELECTING THE ORDER'
      WRITE(6,*) '     BY CHANGING SPECIFICATION FREQUENCIES TO THE'
      WRITE(6,*) '     CORRESPONDING DIGITAL PROTOTYPE FREQUENCIES '
      WRITE(6,*) '       NOTE: YOU WILL NEED PROTOTYPE FREQUENCY    '
      WRITE(6,*) '     RESPONSE PLOTS LIKE THOSE GIVEN IN REF(2) FOR  '
      WRITE(6,*) '     BUTTERWORTH FILTERS (FIG 10.33, PGS 691&692)  '
      WRITE(6,*) '     AND CHEBYSHEV FILTERS (FIG 10.34, PGS 696-698) '
      READ(5,*) ORDER
      IF((TYPE.GT.4).AND.(TYPE.LT.8))THEN
       IF(ORDER.EQ.1)THEN
        WRITE(6,*) 'ELLIPTIC FILTERS GIVEN FOR 2ND-9TH ORDER ONLY'
        WRITE(6,*)
        GO TO 10
       ENDIF
      ENDIF
C
      CALL EXCMS ('CLRSCRN')
      WRITE(6,*) 'DO YOU WANT TO ENTER SPECIFICATION FREQUENCIES'
      WRITE(6,*) 'AS ANALOG OR DIGITAL FREQUENCIES?'
      WRITE(6,*) '    1-ANALOG  '
      WRITE(6,*) '    2-DIGITAL '
      WRITE(6,*) '  ** INTEGER 1 OR 2 ** '
      READ(5,*) AFORDF
      CALL EXCMS ('CLRSCRN')
      IF(AFORDF.EQ.1)THEN
        WRITE(6,*) 'WHAT IS THE SAMPLING FREQUENCY IN KHZ?'
        WRITE(6,*) '    ** DECIMAL **'
        READ(5,*) SSFREQ
        SFREQ=DBLE(SSFREQ)
      ENDIF
      CALL EXCMS ('CLRSCRN')
      WRITE(6,*)
      WRITE(6,*) 'WHAT TYPE OF PASSBAND DUE YOU WANT?'
      WRITE(6,*) '    1-LOWPASS'
      WRITE(6,*) '    2-HIGHPASS'
      WRITE(6,*) '    3-BANDPASS'
      WRITE(6,*) '    4-BANDSTOP'
      WRITE(6,*)
      WRITE(6,*) '        ** INTEGER 1-4 ** '
      READ(5,*) PASS
C     FORMULAS FOR HIGHPASS AND LOWPASS
      IF((PASS.EQ.1).OR.(PASS.EQ.2))THEN
       CALL EXCMS ('CLRSCRN')
       IF(AFORDF.EQ.1)THEN
         WRITE(6,*) 'WHAT IS THE CRITICAL FREQUENCY IN KHZ? '
         WRITE(6,*) '    - NEG 3DB PT FOR BUTTERWORTH  '
         WRITE(6,*) '    - RIPPLE EDGE FOR ELLIPTIC AND CHEBYSHEV'
         WRITE(6,*) '        ** DECIMAL ** '
         READ(5,*) SAFC
         AFC=DBLE(SAFC)
         DFC=(AFC/SFREQ)*PI*(2.0D+00)
       ENDIF
       IF(AFORDF.EQ.2)THEN
         WRITE(6,*) 'WHAT IS THE CRITICAL DIGITAL FREQUENCY?'
```

```
         WRITE(6,*) '     - NEG 3DB FOR BUTTERWORTH '
         WRITE(6,*) '     - RIPPLE EDGE FOR ELLIPTIC AND CHEBYSHEV'
         WRITE(6,*) '       ** DECIMAL ** '
         READ(5,*) SDFC
         DFC=DBLE(SDFC)
       ENDIF
       DFCD2=DFC/2.0D+00
C     CALCULATE THE APPROPRIATE VALUES OF THE DESIGN CONSTANT ALPHA
C     FROM TABLE 10.7 REFERENCE (2) AND REFERENCE (3)
       IF(PASS.EQ.1)THEN
        ALPHA=DSIN((PID4)-(DFCD2))/DSIN((PID4)+(DFCD2))
       ENDIF
       IF(PASS.EQ.2)THEN
        ALPHA=DCOS((PID4)-(DFCD2))/DCOS((PID4)+(DFCD2))
       ENDIF
C     ****************************************************************
C     OPTION TO SHOW CALCULATED VALUE OF ALPHA
       CALL EXCMS ('CLRSCRN')
       WRITE(6,*) 'DO YOU WISH TO SEE THE CALCULATED VALUE OF ALPHA?'
       WRITE(6,*) ' ** 1-YES/2-NO **'
       READ(5,*) LOOK
       IF(LOOK.EQ.1)THEN
        WRITE(6,*)
        WRITE(6,903) ALPHA
        WRITE(6,*)
        WRITE(6,*) '** ENTER 1 TO CONTINUE **'
        READ(5,*) LOOK
       ENDIF
      ENDIF
C     ****************************************************************
C     BANDPASS AND BANDSTOP FORMULAS
      IF((PASS.EQ.3).OR.(PASS.EQ.4))THEN
       CALL EXCMS ('CLRSCRN')
       IF(AFORDF.EQ.1)THEN
         WRITE(6,*) 'WHAT IS THE LOWER CRITICAL FREQUENCY IN KHZ?'
         WRITE(6,*) '    - NEG 3DB FOR BUTTERWORTH'
         WRITE(6,*) '    - RIPPLE EDGE FOR ELLIPTIC AND CHEBYSHEV  '
         WRITE(6,*) '       ** DECIMAL **'
         READ(5,*)  SALC
         ALC=DBLE(SALC)
         CALL EXCMS ('CLRSCRN')
         WRITE(6,*) 'WHAT IS THE UPPER CRITICAL FREQUENCY IN KHZ?'
         WRITE(6,*) '    - NEG 3DB FOR BUTTERWORTH'
         WRITE(6,*) '    - RIPPLE EDGE FOR ELLIPTIC AND CHEBYSHEV '
         WRITE(6,*) '        ** DECIMAL **'
         READ(5,*) SAUC
         AUC=DBLE(SAUC)
         DLC=(ALC/SFREQ)*(2.0D+00)*PI
         DUC=(AUC/SFREQ)*(2.0D+00)*PI
       ENDIF
       IF(AFORDF.EQ.2)THEN
         WRITE(6,*) 'WHAT IS THE LOWER CRITICAL DIGITAL FREQUENCY?'
         WRITE(6,*) '    - NEG 3DB FOR BUTTERWORTH '
         WRITE(6,*) '    - RIPPLE EDGE FOR CHEBYSHEV AND ELLIPTIC'
         WRITE(6,*) '    ** DECIMAL ** '
         READ(5,*) SDLC
         DLC=DBLE(SDLC)
         CALL EXCMS ('CLRSCRN')
         WRITE(6,*) ' WHAT IS THE UPPER CRITICAL DIGITAL FREQUENCY?'
         WRITE(6,*) '    - NEG 3DB FOR BUTTERWORTH '
         WRITE(6,*) '    - RIPPLE EDGE FOR CHEBYSHEV AND ELLIPTIC'
         WRITE(6,*) '    ** DECIMAL ** '
         READ(5,*) SDUC
         DUC=DBLE(SDUC)
       ENDIF
       DLCD2=DLC/2.0D+00
       DUCD2=DUC/2.0D+00
C     CALCULATE APPROPRIATE VALUES OF DESIGN CONSTANTS ALPHA AND K
C     FROM TABLE 10.7 REFERENCE (2) AND REFERENCE (3)
       ALPHA=DCOS((DUCD2)+(DLCD2))/DCOS((DUCD2)-(DLCD2))
       IF(PASS.EQ.4)THEN
        RK=DTAN(PID4)*DTAN((DUCD2)-(DLCD2))
```

```
         B=((2.0D+00)*ALPHA)/(RK+1.0D+00)
         C=((1.0D+00)-RK)/((1.0D+00)+RK)
        ENDIF
        IF(PASS.EQ.3)THEN
         RK=DTAN(PID4)/DTAN((DUCD2)-(DLCD2))
         B=((2.0D+00)*ALPHA*RK)/(RK+(1.0D+00))
         C=(RK-(1.0D+00))/(RK+(1.0D+00))
        ENDIF
C      OPTION TO SHOW CALCULATED VALUE OF ALPHA AND K
        CALL EXCMS ('CLRSCRN')
        WRITE(6,*) 'DO YOU WISH TO SEE THE CALCULATED VALUE OF ALPHA '
        WRITE(6,*) '  AND K?  '
        WRITE(6,*) '      ** 1-YES/2-NO **'
        READ(5,*) LOOK
        IF(LOOK.EQ.1)THEN
         WRITE(6,*)
         WRITE(6,903) ALPHA
         WRITE(6,904) RK
         WRITE(6,*)
         WRITE(6,*) '** ENTER 1 TO CONTINUE **'
         READ(5,*) LOOK
        ENDIF
       ENDIF
C      ************************************************************
C      OPTION TO ASSIST IN DETERMINING THE PROTOTYPE ORDER BY CHANGING
C      SPECIFICATION FREQUENCIES TO APPROPRIATE PROTOTYPE FREQUENCIES
C      FROM EQN 10.274 REFERENCE(1)
       IF(ORDER.EQ.0)THEN
16      CALL EXCMS ('CLRSCRN')
        IF(AFORDF.EQ.1)THEN
          WRITE(6,*) 'WHAT IS THE ANALOG SPECIFICATION FREQUENCY (KHZ)'
          WRITE(6,*) 'THAT YOU WANT TO CONVERT TO A DIGITAL'
          WRITE(6,*) 'LOWPASS PROTOTYPE FREQUENCY?'
          WRITE(6,*) '  ** DECIMAL **'
          READ(5,*) F
          DF=(2.0)*SPI*(F/SSFREQ)
        ENDIF
        IF(AFORDF.EQ.2)THEN
          WRITE(6,*) 'WHAT IS THE DIGITAL SPECIFICATION FREQUENCY'
          WRITE(6,*) 'THAT YOU WANT TO CONVERT TO A DIGITAL'
          WRITE(6,*) 'LOWPASS PROTOTYPE FREQUENCY?'
          WRITE(6,*) '  ** DECIMAL **'
          READ(5,*) DF
        ENDIF
        X=COS(DF)
        Y=SIN(DF)
        Z=CMPLX(X,Y)
        IF(PASS.EQ.1)THEN
         AS=REAL(ALPHA)
         NEWZ=(Z-AS)/((1.0)-(AS*Z))
        ENDIF
        IF(PASS.EQ.2)THEN
         AS=REAL(ALPHA)
         NEWZ=(-1.0)*((Z-AS)/((1.0)-(AS*Z)))
        ENDIF
        IF(PASS.EQ.3)THEN
         BS=REAL(B)
         CS=REAL(C)
         NEWZ=(-1.0)*(((Z**2)-(BS*Z)+CS)/((1.0)-(BS*Z)+(CS*(Z**2))))
        ENDIF
        IF(PASS.EQ.4)THEN
         BS=REAL(B)
         CS=REAL(C)
         NEWZ=((Z**2)-(BS*Z)+CS)/((1.0)-(BS*Z)+(CS*(Z**2)))
        ENDIF
        PTHETA=ABS(ATAN(AIMAG(NEWZ)/REAL(NEWZ)))
        IF(PASS.EQ.1)THEN
         IF(F.GT.SAFC)THEN
          PTHETA=SPI-PTHETA
         ENDIF
        ENDIF
        IF(PASS.EQ.2)THEN
```

FILE: DFCADD FORTRAN A

```
      IF(F.LT.SAFC)THEN
       PTHETA=SPI-PTHETA
      ENDIF
     ENDIF
     IF(PASS.EQ.3)THEN
      IF((F.LT.SALC).OR.(F.GT.SAUC))THEN
       PTHETA=SPI-PTHETA
      ENDIF
     ENDIF
     IF(PASS.EQ.4)THEN
      IF((F.GT.SALC).AND.(F.LT.SAUC))THEN
       PTHETA=SPI-PTHETA
      ENDIF
     ENDIF
     WRITE(6,905) PTHETA
     WRITE(6,*)
     WRITE(6,*) 'WANT TO CONVERT ANOTHER FREQUENCY?'
     WRITE(6,*) '  ** 1-YES/2-NO **'
     READ(5,*) LOOK
     IF(LOOK.EQ.1)GO TO 16
     IF(LOOK.EQ.2)THEN
      WRITE(6,*)
      WRITE(6,*) 'WHAT ORDER PROTOTYPE DO YOU WANT TO USE?'
      WRITE(6,*) '  ** INTEGER 1-5 ** '
      READ(5,*) ORDER
     ENDIF
    ENDIF
C   ****************************************************************
C   LOAD APROT MATRIX WITH APPROPRIATE COEFFICIENTS
C   APROT(TYPE=1) ARE BUTTERWORTH FILTERS
    IF(TYPE.EQ.1)THEN
     IF(ORDER.EQ.1)THEN
      APROT(1,0)=1.0D+00
      APROT(2,0)=1.0D+00
      APROT(2,1)=1.0D+00
     ELSEIF(ORDER.EQ.2)THEN
      APROT(1,0)=1.0D+00
      APROT(2,0)=1.0D+00
      APROT(2,1)=DSQRT(2.0D+00)
      APROT(2,2)=1.0D+00
     ELSEIF(ORDER.EQ.3)THEN
      APROT(1,0)=1.0D+00
      APROT(2,0)=1.0D+00
      APROT(2,1)=2.0D+00
      APROT(2,2)=2.0D+00
      APROT(2,3)=1.0D+00
     ELSEIF(ORDER.EQ.4)THEN
      APROT(1,0)=1.0D+00
      APROT(2,0)=1.0D+00
      APROT(2,1)=2.613126D+00
      APROT(2,2)=3.414214D+00
      APROT(2,3)=2.613126D+00
      APROT(2,4)=1.0D+00
     ELSEIF(ORDER.EQ.5)THEN
      APROT(1,0)=1.0D+00
      APROT(2,0)=1.0D+00
      APROT(2,1)=3.236068D+00
      APROT(2,2)=5.236068D+00
      APROT(2,3)=5.236068D+00
      APROT(2,4)=3.236068D+00
      APROT(2,5)=1.0D+00
     ENDIF
    ENDIF
C   APROT(TYPE=2) ARE CHEBYSHEV FILTERS WITH .5DB RIPPLE
    IF(TYPE.EQ.2)THEN
     IF(ORDER.EQ.1)THEN
      APROT(1,0)=2.862775D+00
      APROT(2,0)=2.862775D+00
      APROT(2,1)=1.0D+00
     ELSEIF(ORDER.EQ.2)THEN
      APROT(1,0)=1.431388D+00
      APROT(2,0)=1.516203D+00
```

```
          APROT(2,1)=1.425625D+00
          APROT(2,2)=1.0D+00
         ELSEIF(ORDER.EQ.3)THEN
          APROT(1,0)=0.715694D+00
          APROT(2,0)=0.715694D+00
          APROT(2,1)=1.534895D+00
          APROT(2,2)=1.252913D+00
          APROT(2,3)=1.0D+00
         ELSEIF(ORDER.EQ.4)THEN
          APROT(1,0)=0.357847D+00
          APROT(2,0)=0.379051D+00
          APROT(2,1)=1.025455D+00
          APROT(2,2)=1.716866D+00
          APROT(2,3)=1.197386D+00
          APROT(2,4)=1.0D+00
         ELSEIF(ORDER.EQ.5)THEN
          APROT(1,0)=0.178923D+00
          APROT(2,0)=0.178923D+00
          APROT(2,1)=0.752518D+00
          APROT(2,2)=1.309575D+00
          APROT(2,3)=1.937368D+00
          APROT(2,4)=1.172491D+00
          APROT(2,5)=1.0D+00
         ENDIF
        ENDIF
C       APROT(TYPE=3) ARE CHEBYSHEV FILTERS WITH 1DB RIPPLE
        IF(TYPE.EQ.3)THEN
         IF(ORDER.EQ.1)THEN
          APROT(1,0)=1.196523D+00
          APROT(2,0)=1.196523D+00
          APROT(2,1)=1.0D+00
         ELSEIF(ORDER.EQ.2)THEN
          APROT(1,0)=0.982613D+00
          APROT(2,0)=1.102510D+00
          APROT(2,1)=1.097734D+00
          APROT(2,2)=1.0D+00
         ELSEIF(ORDER.EQ.3)THEN
          APROT(1,0)=0.491307D+00
          APROT(2,0)=0.491307D+00
          APROT(2,1)=1.238409D+00
          APROT(2,2)=0.988341D+00
          APROT(2,3)=1.0D+00
         ELSEIF(ORDER.EQ.4)THEN
          APROT(1,0)=0.245653D+00
          APROT(2,0)=0.275628D+00
          APROT(2,1)=0.742619D+00
          APROT(2,2)=1.453925D+00
          APROT(2,3)=0.952811D+00
          APROT(2,4)=1.0D+00
         ELSEIF(ORDER.EQ.5)THEN
          APROT(1,0)=0.122827D+00
          APROT(2,0)=0.122827D+00
          APROT(2,1)=0.580534D+00
          APROT(2,2)=0.974396D+00
          APROT(2,3)=1.688816D+00
          APROT(2,4)=0.936820D+00
          APROT(2,5)=1.0D+00
         ENDIF
        ENDIF
C       APROT(TYPE=4) ARE CHEBYSHEV FILTERS WITH 2DB RIPPLE
        IF(TYPE.EQ.4)THEN
         IF(ORDER.EQ.1)THEN
          APROT(1,0)=1.307560D+00
          APROT(2,0)=1.307560D+00
          APROT(2,1)=1.0D+00
         ELSEIF(ORDER.EQ.2)THEN
          APROT(1,0)=0.505803D+00
          APROT(2,0)=0.636768D+00
          APROT(2,1)=0.803816D+00
          APROT(2,2)=1.0D+00
         ELSEIF(ORDER.EQ.3)THEN
          APROT(1,0)=0.326890D+00
```

```
         APROT(2,0)=0.326890D+00
         APROT(2,1)=1.022190D+00
         APROT(2,2)=0.737822D+00
         APROT(2,3)=1.0D+00
        ELSEIF(ORDER.EQ.4)THEN
         APROT(1,0)=0.163445D+00
         APROT(2,0)=0.205765D+00
         APROT(2,1)=0.516798D+00
         APROT(2,2)=1.256482D+00
         APROT(2,3)=0.716215D+00
         APROT(2,4)=1.0D+00
        ELSEIF(ORDER.EQ.5)THEN
         APROT(1,0)=0.081723D+00
         APROT(2,0)=0.081723D+00
         APROT(2,1)=0.459349D+00
         APROT(2,2)=0.693477D+00
         APROT(2,3)=1.499543D+00
         APROT(2,4)=0.706461D+00
         APROT(2,5)=1.0D+00
        ENDIF
       ENDIF
C      APROT(TYPE=5) ARE ELLIPTIC FILTERS
C      ELLIPTIC COEFFICIENTS ARE CALCULATED USING A PROGRAM DEVELOPED
C      BY PROFESSOR D. E. KIRK, NAVAL POSTGRADUATE SCHOOL, MONTEREY,
C      CALIFORNIA  93943
       A02=0.0D+00
       B02=0.0D+00
       B12=0.0D+00
       S0=0.0D+00
       IF(TYPE.EQ.5)THEN
         CALL EXCMS ('CLRSCRN')
         WRITE(6,*) 'WHAT PASSBAND RIPPLE DO YOU WANT? (IN DB)'
         WRITE(6,*) '*** MUST BE EITHER 0.5, 1.0, 2.0, OR 3.0 ***'
         READ(5,*) ASUBP
         WRITE(6,*)
         WRITE(6,*)
         WRITE(6,*) 'WHAT STOPBAND ATTENUATION DO YOU WANT? (IN DB)'
         WRITE(6,*) '*** MUST BE -20,-30,-40,-50,-60, OR -70 ***'
         WRITE(6,*) '*** DO NOT INCLUDE A DECIMAL ***'
         READ(5,*) NATTN
         NATTN=-1*NATTN
         ASUBA=REAL(NATTN)
         IF(ASUBP.EQ.0.5)THEN
           IF(NATTN.EQ.20)THEN
             IF(ORDER.EQ.2)SR=2.76261
             IF(ORDER.EQ.3)SR=1.42189
             IF(ORDER.EQ.4)SR=1.13188
             IF(ORDER.EQ.5)SR=1.04465
             IF(ORDER.EQ.6)SR=1.01553
           ENDIF
           IF(NATTN.EQ.30)THEN
             IF(ORDER.EQ.2)SR=4.80880
             IF(ORDER.EQ.3)SR=1.92322
             IF(ORDER.EQ.4)SR=1.32446
             IF(ORDER.EQ.5)SR=1.12912
             IF(ORDER.EQ.6)SR=1.05394
           ENDIF
           IF(NATTN.EQ.40)THEN
             IF(ORDER.EQ.2)SR=8.848925
             IF(ORDER.EQ.3)SR=2.71147
             IF(ORDER.EQ.4)SR=1.62842
             IF(ORDER.EQ.5)SR=1.27264
             IF(ORDER.EQ.6)SR=1.12697
           ENDIF
           IF(NATTN.EQ.50)THEN
             IF(ORDER.EQ.2)SR=15.06069
             IF(ORDER.EQ.3)SR=3.90430
             IF(ORDER.EQ.4)SR=2.06924
             IF(ORDER.EQ.5)SR=1.48469
             IF(ORDER.EQ.6)SR=1.24101
           ENDIF
           IF(NATTN.EQ.60)THEN
```

```
      IF(ORDER.EQ.2)SR=26.76231
      IF(ORDER.EQ.3)SR=5.67937
      IF(ORDER.EQ.4)SR=2.88325
      IF(ORDER.EQ.5)SR=1.77664
      IF(ORDER.EQ.6)SR=1.41138
    ENDIF
    IF(NATTN.EQ.70)THEN
      IF(ORDER.EQ.2)SR=47.58853
      IF(ORDER.EQ.3)SR=8.38124
      IF(ORDER.EQ.4)SR=3.52135
      IF(ORDER.EQ.5)SR=2.16377
      IF(ORDER.EQ.6)SR=1.61413
    ENDIF
  ENDIF
  IF(ASUBP.EQ.1.0)THEN
    IF(NATTN.EQ.20)THEN
      IF(ORDER.EQ.2)SR=2.32474
      IF(ORDER.EQ.3)SR=1.34797
      IF(ORDER.EQ.4)SR=1.19109
      IF(ORDER.EQ.5)SR=1.12826
      IF(ORDER.EQ.6)SR=1.09902
    ENDIF
    IF(NATTN.EQ.30)THEN
      IF(ORDER.EQ.2)SR=4.00423
      IF(ORDER.EQ.3)SR=1.73254
      IF(ORDER.EQ.4)SR=1.23141
      IF(ORDER.EQ.5)SR=1.19984
      IF(ORDER.EQ.6)SR=1.13799
    ENDIF
    IF(NATTN.EQ.40)THEN
      IF(ORDER.EQ.2)SR=7.14488
      IF(ORDER.EQ.3)SR=2.42619
      IF(ORDER.EQ.4)SR=1.50349
      IF(ORDER.EQ.5)SR=1.28868
      IF(ORDER.EQ.6)SR=1.19887
    ENDIF
    IF(NATTN.EQ.50)THEN
      IF(ORDER.EQ.2)SR=12.48470
      IF(ORDER.EQ.3)SR=3.46761
      IF(ORDER.EQ.4)SR=2.00119
      IF(ORDER.EQ.5)SR=1.40110
      IF(ORDER.EQ.6)SR=1.29851
    ENDIF
    IF(NATTN.EQ.60)THEN
      IF(ORDER.EQ.2)SR=22.17688
      IF(ORDER.EQ.3)SR=5.02120
      IF(ORDER.EQ.4)SR=2.46179
      IF(ORDER.EQ.5)SR=1.67161
      IF(ORDER.EQ.6)SR=1.34334
    ENDIF
    IF(NATTN.EQ.70)THEN
      IF(ORDER.EQ.2)SR=39.42283
      IF(ORDER.EQ.3)SR=7.30093
      IF(ORDER.EQ.4)SR=3.21912
      IF(ORDER.EQ.5)SR=2.02567
      IF(ORDER.EQ.6)SR=1.55847
    ENDIF
  ENDIF
  IF(ASUBP.EQ.2.0)THEN
    IF(NATTN.EQ.20)THEN
      IF(ORDER.EQ.2)SR=1.84332
      IF(ORDER.EQ.3)SR=1.21801
      IF(ORDER.EQ.4)SR=1.13369
      IF(ORDER.EQ.5)SR=1.11567
      IF(ORDER.EQ.6)SR=1.10447
    ENDIF
    IF(NATTN.EQ.30)THEN
      IF(ORDER.EQ.2)SR=3.28235
      IF(ORDER.EQ.3)SR=1.35691
      IF(ORDER.EQ.4)SR=1.18281
      IF(ORDER.EQ.5)SR=1.16394
      IF(ORDER.EQ.6)SR=1.12461
```

```
  ENDIF
  IF(NATTN.EQ.40)THEN
    IF(ORDER.EQ.2)SR=5.76107
    IF(ORDER.EQ.3)SR=2.13923
    IF(ORDER.EQ.4)SR=1.40842
    IF(ORDER.EQ.5)SR=1.16811
    IF(ORDER.EQ.6)SR=1.07316
  ENDIF
  IF(NATTN.EQ.50)THEN
    IF(ORDER.EQ.2)SR=10.19181
    IF(ORDER.EQ.3)SR=3.04137
    IF(ORDER.EQ.4)SR=1.75285
    IF(ORDER.EQ.5)SR=1.33243
    IF(ORDER.EQ.6)SR=1.15868
  ENDIF
  IF(NATTN.EQ.60)THEN
    IF(ORDER.EQ.2)SR=18.09398
    IF(ORDER.EQ.3)SR=4.39729
    IF(ORDER.EQ.4)SR=2.24440
    IF(ORDER.EQ.5)SR=1.56860
    IF(ORDER.EQ.6)SR=1.28693
  ENDIF
  IF(NATTN.EQ.70)THEN
    IF(ORDER.EQ.2)SR=32.15951
    IF(ORDER.EQ.3)SR=6.40894
    IF(ORDER.EQ.4)SR=2.92363
    IF(ORDER.EQ.5)SR=1.88906
    IF(ORDER.EQ.6)SR=1.46330
  ENDIF
ENDIF
IF(ASUBP.EQ.3.0)THEN
  IF(NATTN.EQ.20)THEN
    IF(ORDER.EQ.2)SR=1.73915
    IF(ORDER.EQ.3)SR=1.15516
    IF(ORDER.EQ.4)SR=1.03853
    IF(ORDER.EQ.5)SR=1.00996
    IF(ORDER.EQ.6)SR=1.00260
  ENDIF
  IF(NATTN.EQ.30)THEN
    IF(ORDER.EQ.2)SR=2.90352
    IF(ORDER.EQ.3)SR=1.45814
    IF(ORDER.EQ.4)SR=1.14542
    IF(ORDER.EQ.5)SR=1.05020
    IF(ORDER.EQ.6)SR=1.01783
  ENDIF
  IF(NATTN.EQ.40)THEN
    IF(ORDER.EQ.2)SR=5.05584
    IF(ORDER.EQ.3)SR=1.98022
    IF(ORDER.EQ.4)SR=1.34663
    IF(ORDER.EQ.5)SR=1.13934
    IF(ORDER.EQ.6)SR=1.05892
  ENDIF
  IF(NATTN.EQ.50)THEN
    IF(ORDER.EQ.2)SR=8.93009
    IF(ORDER.EQ.3)SR=2.79862
    IF(ORDER.EQ.4)SR=1.66148
    IF(ORDER.EQ.5)SR=1.28850
    IF(ORDER.EQ.6)SR=1.13534
  ENDIF
  IF(NATTN.EQ.60)THEN
    IF(ORDER.EQ.2)SR=15.84610
    IF(ORDER.EQ.3)SR=4.03471
    IF(ORDER.EQ.4)SR=2.11596
    IF(ORDER.EQ.5)SR=1.50711
    IF(ORDER.EQ.6)SR=1.25325
  ENDIF
  IF(NATTN.EQ.70)THEN
    IF(ORDER.EQ.2)SR=28.15950
    IF(ORDER.EQ.3)SR=5.87251
    IF(ORDER.EQ.4)SR=2.74749
    IF(ORDER.EQ.5)SR=1.80680
    IF(ORDER.EQ.6)SR=1.41799
```

```
          ENDIF
        ENDIF
        SPI=3.14159
        RK=1.0/SR
        N=ORDER
        EN=N
        KPR=SQRT(1.0-RK**2)
        Q0=0.5*(1.0-SQRT(KPR))/(1.0+SQRT(KPR))
        Q=Q0+2.0*Q0**5+15.0*Q0**9+150.0*Q0**13
        D=(10.0**(0.1*ASUBA)-1.0)/(10.0**(0.1*ASUBP)-1.0)
        LAM=(1.0/(2.0*EN))*ALOG((10.0**(0.05*ASUBP)+1.0)/(10.0**(0.05
     C    *ASUBP)-1.0))
        ISTOP=0
        DENSUM=0.0
        ENSUM=SINH(LAM)
        M=1
17      EM=M
        ENUM=((-1.0)**M)*(Q**(M*(M+1)))*SINH((2.0*EM+1.0)*LAM)
        DEN=2.0*((-1.0)**M)*(Q**(M**2))*COSH(2.0*EM*LAM)
        ENSUM=ENSUM+ENUM
        DENSUM=DENSUM+DEN
        IF(M.GE.2)THEN
          RN=ABS(ENUM/ENSUM)
          RD=ABS(DEN/(1.0+DENSUM))
          IF((RN.LE.1.0E-8).AND.(RD.LE.1.0E-8))THEN
            ISTOP=1
          ENDIF
        ENDIF
        M=M+1
        IF(ISTOP.EQ.0)THEN
          GO TO 17
        ENDIF
        SIGMA0=ABS((2.0*(Q**0.25)*ENSUM)/(1.0+DENSUM))
        W=SQRT((1.0+RK*(SIGMA0**2))*(1.0+(SIGMA0**2/RK)))
        IN=MOD(N,2)
        IF(IN.EQ.0)THEN
          IR=N/2
        ELSE
          IR=(N-1)/2
        ENDIF
        DO 18 I=1,IR
          IF(IN.EQ.0)THEN
            EMU=I-0.5
          ELSE
            EMU=I
          ENDIF
          ISTOP=0
          DENSUM=0.0
          ENSUM=SIN(PI*EMU/EN)
          M=1
19        EM=M
          ENUM=((-1.0)**M)*(Q**(M*(M+1)))*SIN((2.0*EM+1.0)*PI*EMU/EN)
          DEN=2.0*((-1.0)**M)*(Q**(M**2))*COS(2.0*EM*PI*EMU/EN)
          ENSUM=ENSUM+ENUM
          DENSUM=DENSUM+DEN
          IF(M.GE.2)THEN
            RN=ABS(ENUM/ENSUM)
            RD=ABS(DEN/(1.0+DENSUM))
            IF((RN.LE.1.0E-8).AND.(RD.LE.1.0E-8))THEN
              ISTOP=1
            ENDIF
          ENDIF
          M=M+1
          IF(ISTOP.EQ.0)THEN
            GO TO 19
          ENDIF
          OMEGA(I)=(2.0*(Q**0.25)*ENSUM)/(1.0+DENSUM)
          V(I)=SQRT((1.0-RK*OMEGA(I)**2)*(1.0-(OMEGA(I)**2/RK)))
          A0(I)=1.0/(OMEGA(I)**2)
          DN=1.0+(SIGMA0*OMEGA(I))**2
          B0(I)=((SIGMA0*V(I))**2+(OMEGA(I)*W)**2)/(DN**2)
          B1(I)=(2.0*SIGMA0*V(I))/DN
```

```
18      CONTINUE
        H0=1.0
        IF(IN.EQ.0)THEN
          DO 20 I=1,IR
            H0=H0*B0(I)/A0(I)
20        CONTINUE
          H0=H0*(10.0**(-0.05*ASUBP))
        ELSE
          DO 21 I=1,IR
            H0=H0*B0(I)/A0(I)
21        CONTINUE
          H0=H0*SIGMA0
        ENDIF
        R=DBLE(SR)
        H0=DBLE(H0)
        A01=DBLE(A0(1))
        B01=DBLE(B0(1))
        B11=DBLE(B1(1))
        IF((ORDER.EQ.2).OR.(ORDER.EQ.3))THEN
          APROT(1,0)=H0*A01
          APROT(1,1)=0.0D+00
          APROT(1,2)=H0
          APROT(1,3)=0.0D+00
          IF(ORDER.EQ.2)THEN
            APROT(2,0)=B01
            APROT(2,1)=B11
            APROT(2,2)=1.0D+00
          ENDIF
          IF(ORDER.EQ.3)THEN
            S0=DBLE(SIGMA0)
            APROT(2,0)=S0*B01
            APROT(2,1)=B01+(B11*S0)
            APROT(2,2)=S0+B11
            APROT(2,3)=1.0D+00
          ENDIF
        ENDIF
        IF((ORDER.EQ.4).OR.(ORDER.EQ.5))THEN
          A02=DBLE(A0(2))
          B02=DBLE(B0(2))
          B12=DBLE(B1(2))
          APROT(1,0)=H0*A01*A02
          APROT(1,1)=0.0D+00
          APROT(1,2)=H0*(A01+A02)
          APROT(1,3)=0.0D+00
          APROT(1,4)=H0
          APROT(1,5)=0.0D+00
          IF(ORDER.EQ.4)THEN
            APROT(2,0)=B01*B02
            APROT(2,1)=(B11*B02)+(B01*B12)
            APROT(2,2)=B01+B02+(B11*B12)
            APROT(2,3)=B11+B12
            APROT(2,4)=1.0D+00
          ENDIF
          IF(ORDER.EQ.5)THEN
            S0=DBLE(SIGMA0)
            APROT(2,0)=S0*B01*B02
            APROT(2,1)=(S0*((B11*B02)+(B01*B12)))+(B01*B02)
            APROT(2,2)=(S0*(B01+B02+(B11*B12)))+(B11*B02)+(B01*B12)
            APROT(2,3)=(S0*(B11+B12))+B01+B02+(B11*B12)
            APROT(2,4)=S0+B11+B12
            APROT(2,5)=1.0D+00
          ENDIF
        ENDIF
        IF((ORDER.EQ.6).OR.(ORDER.EQ.7))THEN
          A02=DBLE(A0(2))
          A03=DBLE(A0(3))
          B02=DBLE(B0(2))
          B03=DBLE(B0(3))
          B12=DBLE(B1(2))
          B13=DBLE(B1(3))
          APROT(1,0)=H0*A01*A02*A03
          APROT(1,1)=0.0D+00
```

```
          APROT(1,2)=H0*((A02*A03)+(A01*(A02+A03)))
          APROT(1,3)=0.0D+00
          APROT(1,4)=H0*(A01+A02+A03)
          APROT(1,5)=0.0D+00
          APROT(1,6)=H0
          IF((ORDER.EQ.6).OR.(ORDER.EQ.7))THEN
            APROT(2,0)=B01*B02*B03
            APROT(2,1)=(B11*B02*B03)+(B01*((B02*B13)+(B12*B03)))
            APROT(2,2)=(B02*B03)+(B11*((B02*B13)+(B12*B03)))+
     C         (B01*((B02+B03)+(B12*B13)))
            APROT(2,3)=B01*(B12+B13)+(B11*((B02+B03)+(B12*B13)))+
     C         ((B02*B13)+(B12*B03))
            APROT(2,4)=B01+(B11*(B12+B13))+((B02+B03)+(B12*B13))
            APROT(2,5)=B11+B12+B13
            APROT(2,6)=1.0D+00
          ENDIF
          IF(ORDER.EQ.7)THEN
            S0=DBLE(SIGMA0)
            APROT(2,7)=1.0D+00
            APROT(2,6)=(S0*APROT(2,6))+APROT(2,5)
            APROT(2,5)=(S0*APROT(2,5))+APROT(2,4)
            APROT(2,4)=(S0*APROT(2,4))+APROT(2,3)
            APROT(2,3)=(S0*APROT(2,3))+APROT(2,2)
            APROT(2,2)=(S0*APROT(2,2))+APROT(2,1)
            APROT(2,1)=(S0*APROT(2,1))+APROT(2,0)
            APROT(2,0)=S0*APROT(2,0)
          ENDIF
        ENDIF
      ENDIF
C     NORMALIZE THE ELLIPTIC PROTOTYPES
      IF(TYPE.EQ.5)THEN
       DO 25 I=1,2
        DO 26 J=0,ORDER
         APROT(I,J)=APROT(I,J)/((DSQRT(R))**J)
26      CONTINUE
25     CONTINUE
      ENDIF
C     APROT(TYPE=6) ALLOWS THE INPUT OF ANY TYPE OF PROTOTYPE FILTER
C       BY INPUTING THE COEFFICIENTS OF S
      IF(TYPE.EQ.6)THEN
       WRITE(6,*)
       WRITE(6,*) 'THIS OPTION ALLOWS YOU TO USE ANY ANALOG  '
       WRITE(6,*) ' PROTOTYPE OF ORDER 1-10         '
       WRITE(6,*) '** COEFFICIENTS MUST BE IN DOUBLE PRECISION '
       WRITE(6,*) '     FORMAT (0.123456789D+00) ** '
       WRITE(6,*)
       WRITE(6,*) 'INPUT THE NUMERATOR COEFFICIENTS FIRST'
       DO 50 I=0,ORDER
        WRITE(6,*) 'WHAT IS THE COEFFICIENT OF S**',I,'?'
        READ(5,*)  APROT(1,I)
50     CONTINUE
       WRITE(6,*)
       CALL EXCMS ('CLRSCRN')
       WRITE(6,*) 'NOW INPUT THE DENOMINATOR COEFFICIENTS'
       WRITE(6,*)
       DO 51 J=0,ORDER
        WRITE(6,*) 'WHAT IS THE COEFFICIENT OF S**',J,'?'
        READ(5,*) APROT(2,J)
51     CONTINUE
      ENDIF
C     ***********************************************************
C     ***********************************************************
C     DEFINE ELEMENTS OF THE BILINEAR TRANSFORMATION MATRIX
C     ***********************************************************
C     FOR ALL THE BILINEAR TRANSFORATION MATRICES, THE FIRST COLUMN
C     HAS ALTERNATING +1'S AND -1'S. THE LAST COLUMN HAS ALL +1'S
      DO 60 I=1,ORDER
       DO 61 J=0,I
        BILINR(I,J,0)=(-1.0)**J
        BILINR(I,J,I)=1.0
61     CONTINUE
60    CONTINUE
```

FILE: DFCADD   FORTRAN A

```
C
C     FIRST ORDER MATRIX HAS ALREADY BEEN LOADED CONTINUE INTERATION
C     UNTIL YOU HAVE THE APPROPRIATE MATRIX
      DO 75 K=2,ORDER
       KM1=K-1
C     LOAD ELEMENTS IN INTERIOR COLUMNS, ALL ROWS EXCEPT LAST ONE
       DO 76 L=0,KM1
        DO 77 M=1,KM1
         MM1=M-1
         BILINR(K,L,M)=BILINR(KM1,L,M)+BILINR(KM1,L,MM1)
77      CONTINUE
76     CONTINUE
C     NOW LOAD THE LAST ROW
       DO 78 N=1,KM1
        BILINR(K,K,N)=BILINR(K,0,N)*((-1.0D+00)**(K+N))
78     CONTINUE
75    CONTINUE
C     ****************************************************************
C     THE FOLLOWING CLEARS DPROT PRIOR TO LOADING
      DO 80 K=1,2
       DO 81 L=0,10
        DPROT(K,L)=0.0
81     CONTINUE
80    CONTINUE
C     ****************************************************************
C     DIGITIZE THE ANALOG PROTOTYPE USING THE BILINEAR TRANSFORMATION
C     ****************************************************************
      DO 100 N=1,2
       DO 101 J=0,ORDER
        SUMR=0.0
        DO 102 I=0,ORDER
         TEMP=APROT(N,I)*BILINR(ORDER,I,J)
         SUMR=SUMR+TEMP
102     CONTINUE
        DPROT(N,J)=SUMR
101    CONTINUE
100   CONTINUE
C     ****************************************************************
C     OPTION TO SHOW DIGITAL PROTOTYPE COEFFICIENTS
      CALL EXCMS ('CLRSCRN')
      WRITE(6,*) 'THE DIGITAL PROTOTYPE HAS BEEN COMPUTED. DO YOU'
      WRITE(6,*) 'WISH TO CHECK THE PROTOTYPE COEFFICIENTS?'
      WRITE(6,*) '  ** 1-YES/2-NO **'
      READ(5,*) LOOK
      IF(LOOK.EQ.1)THEN
C     MAKE DENOMINATOR COEFFICIENT OF Z**ORDER = 1.0
       DO 105 K=1,2
        DO 106 L=0,ORDER
         DPROT(K,L)=DPROT(K,L)/DPROT(2,ORDER)
106     CONTINUE
105    CONTINUE
       CALL EXCMS ('CLRSCRN')
       WRITE(6,*) 'THE DIGITAL PROTOTYPE NUMERATOR COEFFICIENTS ARE:'
       DO 111 N=ORDER,0,-1
        WRITE(6,902) DPROT(1,N),N
111     CONTINUE
       WRITE(6,*)
       WRITE(6,*) 'THE DIGITAL PROTOTYPE DENOMINATOR COEFFICIENTS ARE:'
       DO 112 M=ORDER,0,-1
        WRITE(6,902) DPROT(2,M),M
112    CONTINUE
       WRITE(6,*)
       WRITE(6,*) '** 1 TO CONTINUE **'
       READ(5,*) LOOK
      ENDIF
C     ****************************************************************
C     NOW DO THE DIGITAL TO DIGITAL TRANSFORMATIONS
C     ****************************************************************
      IF((PASS.EQ.1).OR.(PASS.EQ.2))THEN
C     DEFINE ELEMENTS OF THE LOWPASS AND HIGHPASS DIGITAL-DIGITAL
C     TRANSFORMATION MATRIX
C     DEFINE THE ELEMENTS FOR FIRST AND LAST COLUMNS
```

FILE: DFCADD   FORTRAN A

```
        DO 170 I=1,10
         DO 171 J=0,I
          LPHPTR(I,J,0)=(((-1.0)*ALPHA)**J)
          IMJ=I-J
          LPHPTR(I,IMJ,I)=LPHPTR(I,J,0)
171      CONTINUE
170     CONTINUE
C      FIRST ORDER MATRIX HAS BEEN DEFINED ABOVE
C      SECOND TO TENTH ORDER BELOW
        DO 175 K=2,10
         KM1=K-1
         DO 176 L=0,KM1
          DO 177 M=1,KM1
           MM1=M-1
           LPHPTR(K,L,M)=LPHPTR(KM1,L,M)+((LPHPTR(KM1,L,MM1))*(-1.0D+00)
     C*ALPHA)
177       CONTINUE
176      CONTINUE
         DO 178 N=1,KM1
          KMN=K-N
          LPHPTR(K,K,N)=LPHPTR(K,0,KMN)
178      CONTINUE
175     CONTINUE
C      ***************************************************************
C      CALCULATE THE FINAL FILTER
C      FOR HIGHPASS, THE PROTOTYPE MUST SUBSTITUTE (-Z) FOR (Z)
        IF(PASS.EQ.2)THEN
         DO 180 K=1,2
          DO 181 L=0,ORDER
           DPROT(K,L)=DPROT(K,L)*((-1.0)**L)
181       CONTINUE
180      CONTINUE
        ENDIF
        DO 185 K=1,2
         DO 186 L=0,ORDER
          SUMR=0.0D+00
          DO 187 M=0,ORDER
           TEMP=DPROT(K,M)*LPHPTR(ORDER,M,L)
           SUMR=SUMR+TEMP
187       CONTINUE
          DFLTR(K,L)=SUMR
186      CONTINUE
185     CONTINUE
       ENDIF
C      ***************************************************************
C      BANDPASS AND BANDSTOP FORMULAS
       IF((PASS.EQ.3).OR.(PASS.EQ.4))THEN
C      LOAD THE BANDPASS/BANDSTOP TRANSFORMATION MATRIX
C      FIRST LOAD THE FIRST AND LAST COLUMNS
        DO 225 I=1,10
         DO 226 J=0,I
          BPBSTR(I,J,0)=C**J
          I2=2*I
          IMJ=I-J
          BPBSTR(I,IMJ,I2)=BPBSTR(I,J,0)
226      CONTINUE
225     CONTINUE
C      LOAD THE REST OF THE FIRST ORDER MATRIX
        BPBSTR(1,0,1)=(-1.0D+00)*B
        BPBSTR(1,1,1)=(-1.0D+00)*B
C      LOAD THE 2ND-10TH ORDER TRANSFORMATION MATRICES
        DO 230 K=2,10
         K2=2*K
         K2M1=K2-1
         KM1=K-1
         K2M2=K2-2
         KM12=2*KM1
         KM12M1=KM12-1
C      LOAD THE SECOND AND SECOND TO LAST COLUMNS
         DO 231 L=0,KM1
          BPBSTR(K,L,1)=BPBSTR(KM1,L,1)+(BPBSTR(KM1,L,0)*(-1.0D+00)*B)
          BPBSTR(K,0,K2M1)=(BPBSTR(KM1,0,KM12)*(-1.0D+00)*B)+
```

FILE: DFCADD FORTRAN A

```
      C     (BPBSTR(KM1,0,KM12M1)*C)
            KML=K-L
            BPBSTR(K,KML,K2M1)=BPBSTR(K,L,1)
231       CONTINUE
C       NOW LOAD INTERIOR ELEMENTS EXCEPT FOR LAST ROW
          DO 236 L=0,KM1
           DO 237 M=2,K2M2
            MM1=M-1
            MM2=M-2
            BPBSTR(K,L,M)=BPBSTR(KM1,L,M)+(BPBSTR(KM1,L,MM1)*(-1.0D+00)*
      C     B)+(BPBSTR(KM1,L,MM2)*C)
237        CONTINUE
236       CONTINUE
C       NOW FILL THE LAST ROW
          DO 238 N=1,K2M1
           K2MN=K2-N
           BPBSTR(K,K,K2MN)=BPBSTR(K,0,N)
238       CONTINUE
230      CONTINUE
C       ***************************************************************
C       CALCULATE THE FINAL FILTER
C       FOR THE BANDPASS CASE REPLACE (Z) WITH (-Z)
          IF(PASS.EQ.3)THEN
           DO 240 M=1,2
            DO 241 N=0,ORDER
             DPROT(M,N)=DPROT(M,N)*((-1.0D+00)**N)
241         CONTINUE
240        CONTINUE
          ENDIF
         ORDER2=2*ORDER
         DO 245 K=1,2
          DO 246 L=0,ORDER2
           SUMR=0.0D+00
           DO 247 M=0,ORDER
            TEMP=DPROT(K,M)*BPBSTR(ORDER,M,L)
            SUMR=SUMR+TEMP
247        CONTINUE
           DFLTR(K,L)=SUMR
246       CONTINUE
245      CONTINUE
        ENDIF
C       ***************
C       OUTPUT OPTIONS
C       ***************
        IF((PASS.EQ.3).OR.(PASS.EQ.4))THEN
         END=2*ORDER
        ENDIF
        IF((PASS.EQ.1).OR.(PASS.EQ.2))THEN
         END=ORDER
        ENDIF
C       MAKE DENOMINATOR COEFFICIENT OF Z**END EQUAL TO 1
        DO 250 M=1,2
         DO 251 N=0,END
          DFLTR(M,N)=DFLTR(M,N)/DFLTR(2,END)
251      CONTINUE
250     CONTINUE
        CALL EXCMS ('CLRSCRN')
        WRITE(6,*)
        WRITE(6,*) 'THE FINAL DIGITAL FILTER IS:'
        WRITE(6,*)
        WRITE(6,*) '     NUMERATOR COEFFICIENTS:'
        DO 300 I=END,0,-1
         WRITE(6,902) DFLTR(1,I),I
300     CONTINUE
        WRITE(6,*) '     DENOMINATOR COEFFICIENTS:'
        DO 301 J=END,0,-1
         WRITE(6,902) DFLTR(2,J),J
301     CONTINUE
        WRITE(6,*)
        WRITE(6,*) ' ** ENTER 1 TO CONTINUE **'
        READ(5,*) LOOK
C       ***************************************************************
```

FILE: DFCADD FORTRAN A

```
C     OPTION TO SPOT CHECK FILTER OUTPUT MAGNITUDE AND FREQUENCY
      CALL EXCMS ('CLRSCRN')
      WRITE(6,*) 'THE TRANSFER FUNCTION FOR THIS FILTER HAS BEEN'
      WRITE(6,*) 'COMPUTED. DO YOU WANT TO SPOT CHECK THE MAGNITUDE'
      WRITE(6,*) 'AND PHASE OF THE FILTER OUTPUT FOR A GIVEN '
      WRITE(6,*) 'FREQUENCY?'
      WRITE(6,*) '    ** 1-YES/2-NO ** '
      READ(5,*)  LOOK
      IF(LOOK.EQ.1)THEN
310    CALL EXCMS ('CLRSCRN')
       IF(AFORDF.EQ.1)THEN
         WRITE(6,*) 'WHAT ANALOG FREQUENCY(IN KHZ)'
         WRITE(6,*) ' DO YOU WANT TO CHECK?'
         WRITE(6,*) '    ** DECIMAL **'
         READ(5,*)  F
         DF=(F/SSFREQ)*2.0*SPI
       ENDIF
       IF(AFORDF.EQ.2)THEN
         WRITE(6,*) 'WHAT DIGITAL FREQUENCY DO YOU WANT TO CHECK?'
         WRITE(6,*) '  ** DECIMAL ** '
         READ(5,*) DF
       ENDIF
       X=COS(DF)
       Y=SIN(DF)
       Z=CMPLX(X,Y)
       XN=REAL(DFLTR(1,0))
       YN=0.0
       XD=REAL(DFLTR(2,0))
       YD=0.0
       HOFZN=CMPLX(XN,YN)
       HOFZD=CMPLX(XD,YD)
       DO 320 N=1,END
        ZN=REAL(DFLTR(1,N))*(Z**N)
        ZD=REAL(DFLTR(2,N))*(Z**N)
        HOFZN=HOFZN+ZN
        HOFZD=HOFZD+ZD
320    CONTINUE
       HOFZ=HOFZN/HOFZD
       FMAG=SQRT((REAL(HOFZ)**2)+(AIMAG(HOFZ)**2))
       FMAGDB=20.0*ALOG10(FMAG)
       FPHSE=ATAN(AIMAG(HOFZ)/REAL(HOFZ))
       WRITE(6,*) 'THE FILTER OUTPUT WILL BE:'
       WRITE(6,914) FMAGDB
       WRITE(6,915) FPHSE
       WRITE(6,*)
       WRITE(6,*) 'DO YOU WANT TO CHECK ANOTHER FREQUENCY?'
       WRITE(6,*) '    ** 1-YES/2-NO ** '
       READ(5,*) LOOK
       IF(LOOK.EQ.1)GO TO 310
      ENDIF
C     ************************************************************
C     OPTION TO LOAD FREQUENCY RESPONSE INTO A DATA FILE
      CALL EXCMS ('CLRSCRN')
      WRITE(6,*) 'DO YOU WANT THE FILTERS FREQUENCY RESPONSE '
      WRITE(6,*) 'DATA TO BE ENTERED INTO A DATA FILE FOR USE'
      WRITE(6,*) 'BY A PLOTTING PROGRAM? '
      WRITE(6,*) ' NOTE: THE FILE NAME WILL BE FILTER, FILE TYPE'
      WRITE(6,*) '     WILL BE DATA, 100 POINTS WILL BE CALCULATED'
      WRITE(6,*) '     FOR A DIGITAL FREQUENCY FROM 0 TO PI, THE FIRST'
      WRITE(6,*) '     COLUMN WILL BE THE DIGITAL FREQUENCY (EXPRESSED '
      WRITE(6,*) '     AS A FRACTION OF THE SAMPLING FREQUENCY 0-1/2) '
      WRITE(6,*) '     THE SECOND COLUMN WILL BE THE MAGNITUDE, AND'
      WRITE(6,*) '     THE THIRD COLUMN WILL BE THE PHASE(DEGREES)'
      WRITE(6,*) '     ** 1-MAKE DATA FILE/2-DON''T BOTHER **'
      READ(5,*)  LOOK
      IF(LOOK.EQ.1)THEN
       DO 350 I=0,100
        DF=REAL(I)*(5.0/1000.0)
        DFR=REAL(I)*(SPI/100.0)
        X=COS(DFR)
        Y=SIN(DFR)
        Z=CMPLX(X,Y)
```

FILE: DFCADD FORTRAN A

```
        XN=REAL(DFLTR(1,0))
        YN=0.0
        XD=REAL(DFLTR(2,0))
        YD=0.0
        HOFZN=CMPLX(XN,YN)
        HOFZD=CMPLX(XD,YD)
        DO 351 N=1,END
         ZN=REAL(DFLTR(1,N))*(Z**N)
         ZD=REAL(DFLTR(2,N))*(Z**N)
         HOFZN=HOFZN+ZN
         HOFZD=HOFZD+ZD
351     CONTINUE
        HOFZ=HOFZN/HOFZD
        FMAG=SQRT((REAL(HOFZ)**2)+(AIMAG(HOFZ)**2))
        IF(REAL(HOFZ).NE.0.0)THEN
         FPHSER=ATAN(AIMAG(HOFZ)/REAL(HOFZ))
        ELSEIF(REAL(HOFZ).EQ.0.0)THEN
         IF(AIMAG(HOFZ).GT.0.0)FPHSER=1.570796
         IF(AIMAG(HOFZ).LT.0.0)FPHSER=-1.570796
         IF(AIMAG(HOFZ).EQ.0.0)FPHSER=0.0
        ENDIF
        FPHSE=FPHSER*(180.0/SPI)
        WRITE(8,916) DF,FMAG,FPHSE
350    CONTINUE
      ENDIF
C     ****************************************************************
901   FORMAT(5X,F10.5,' S**',I1)
902   FORMAT(5X,F10.5,' Z**',I1)
903   FORMAT(3X,'ALPHA= ',F9.4)
904   FORMAT(3X,'K= ',F9.4)
905   FORMAT(3X,'THE LOWPASS DIGITAL PROTOTYPE FREQ. IS  ',F5.3)
914   FORMAT(3X,'MAGNITUDE =  ',F8.2,' DB')
915   FORMAT(3X,'PHASE =  ',F6.2,' RADIANS')
916   FORMAT(3X,F4.3,10X,F5.3,10X,F6.2)
      STOP
      END
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	1
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
4.	Commandant of the Marine Corps Code TE 06 Headquarters, U.S. Marine Corps Washington, DC 20380-0001	1
5.	United States Military Academy Department of Electrical Engineering ATTN: MAJ E. Siomacco West Point, New York 10996	1
6.	Department of Electrical and Computer Engineering ATTN: Prof. R. Strum, Code 62St Naval Postgraduate School Monterey, California 93943-5000	1
7.	Department of Electrical and Computer Engineering ATTN: Prof. J. England, Code 62 Naval Postgraduate School Monterey, California 93943-5000	1
8.	Department of Electrical and Computer Engineering ATTN: Prof. P. Moose, Code 62Me Naval Postgraduate School Monterey, California 93943-5000	1
9.	Department of Electrical and Computer Engineering ATTN: Prof. D. Kirk, Code 62 Naval Postgraduate School Monterey, California 93943-5000	1

Thesis
R37825 Rich
c.1 Algorithms for computer
aided design of digital
filters.